Virtual Circuit Switching Concept in Ad–Hoc Networking

A thesis
submitted in partial fulfilment
of the requirements for the Degree
of
Master of Science
in the
University of Canterbury
by
Sun Yen-Rong

University of Canterbury
2004

Examining Committee

Supervisor	Professor Krzysztof Pawlikowski
	Department of Computer Science,
	University of Canterbury, New Zealand
Associate Supervisor	Associate Professor Harsha Sirisena
	Department of Electrical & Computer Engineering,
	University of Canterbury, New Zealand
External Examiner	Professor Sergio Palazzo
	University of Catania, Italy

This thesis is dedicated to
everyone who gave me love, friendship and support during my research.

Abstract

This thesis investigates issues of implementing virtual circuit switching in an ad–hoc network. Traditionally, an ad–hoc network uses datagram switching for transmitting a message which is many packets long. Two main challenges for implementing virtual circuit switching in an ad–hoc network are: (1) finding a medium access control (MAC) protocol that supports "virtual circuit" and (2) dealing with the rapid changes of network topology. A major advantage of using virtual circuit switching is its capability to provide Quality of Service during a communication session.

Ad–hoc Virtual Switching Routing (AVSR) protocol is a cross–layered traffic control protocol developed to demonstrate virtual circuit switching in an ad–hoc network. It is a reactive routing protocol running over a self–administrative Time Division Multiple Access (TDMA) MAC protocol. The evaluation of AVSR shows it is applicable to implement virtual circuit switching in an ad–hoc network, however its performance degrades significantly as the number of nodes/terminals in the network increases. The conclusion of this thesis gives recommendations for future research of virtual circuit switching in ad–hoc networks.

Acknowledgements

I would like to thank my supervisors, Professor Krzysztof Pawlikowski and Associate Professor Harsha Sirisena. They both gave me guidelines for doing a scientific research, and ensured the quality of my work. I also want to thank all my friends who have been supporting me over the period of my research. Andreas, Francis, James and Malcolm, thank you for opinions and suggestions which help in my development. Aun, Behshid, Ding, Fong, Hans, Josh, Percy, and both Sungs, thank you for your company when I felt lost and tired for my research. Fiona, thank you for proofreading my unreadable writing. At last, I would like to thank Dad, Mum, Jessie and other members in my family for their generous and kind supports. I would not be able to finish this thesis without you.

Contents

List of Figures

Chapter 1

Introduction

An ad hoc network is a collection of nodes connected by wireless communication channels, forming a temporary network without using a pre–existing infrastructure or centralised administration. Data transfer in ad–hoc networks has usually been based on datagram switching. An alternative technique, known as virtual circuit switching, has not been tried. To consider a feasibility of implementing virtual circuit switching in ad–hoc network networks, we proposed and analysed the performance of a cross–layered switching protocol, named Ad–Hoc Virtual Switching Routing (AVSR). Our research shows the applicability of implementing virtual circuit switching in an ad–hoc network. Furthermore, the results of our evaluation gives significant indications to future developments of virtual circuit switching in ad–hoc networks.

In this chapter, we overview the design issues in ad–hoc networks and discuss the structure of this thesis.

1.1 Design Issue of Ad–Hoc Network

In the world of wireless communication, there are two classes of mobile wireless networks. Networks of the first class uses base stations that are gateways to a fixed and

wired infrastructure backbone network. They are usually referred to as infrastructured networks. A mobile terminal within an infrastructured network communicates with its nearest gateway and a "hand-off" occurs when the terminal leaves the communication radius of its gateway and enters the range of another gateway. A typical example of networks in this class is the Global System of Mobile communication (GSM) network.

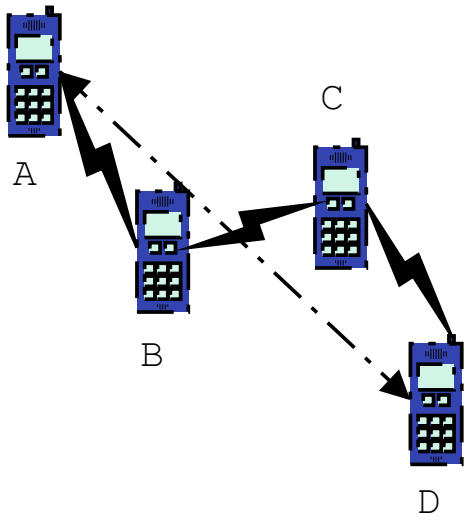


Figure 1.1: A simple ad–hoc network.

The second class of mobile wireless networks have no infrastructure, and are usually referred to as an ad–hoc networks. In such a system, wireless nodes form a temporary network without using any pre–existing and fixed infrastructure. Each node functions as a router itself and is responsible for discovering and maintaining route information to other nodes within the network. Possible uses of ad hoc networking include: emergency search and rescue missions after natural disasters, communication in areas where access is difficult, such as Antarctica, business associates sharing information during a meeting, or soldiers relaying information between themselves on a battlefield [14]. An example of a simple ad–hoc network is shown in Figure 1.1. In this network, no central controller is used. Therefore node A has to use its neighbours, node B or node C, for

forwarding a message to node D if node D is out of the direct communication range of node A.

One major advantage of ad–hoc networks is their fast deployment. The setup of an ad–hoc network is much faster than that of an infrastructure network because no time is required for building infrastructure. In contrast, having a fixed infrastructure in a network makes traffic administration easier. However building infrastructure for a network that has a short lifecycle is not cost effective. The time required to build the infrastructure may be so long that the network service may still not be available even when the need for service ends. In such situations, an ad–hoc network is a better candidate solution.

In wireless networks with infrastructure, fixed controllers track the locations of wireless terminals and provide route information to them. In addition, these controllers can also be responsible for medium access control. A controller can actively assign terminals to different channels for avoiding transmission collision. The absence of infrastructure causes new challenges for traffic management in ad–hoc networks. All traffic management functions have to be distributed over all wireless stations. Solutions that have been proposed to solve this issue are discussed in Chapter 2.

At present, most traffic control solutions proposed for ad–hoc networking are based on datagram switching. In this approach, packets in a communication session are treated as independent entities. The major advantage of datagram switching is that there is no initial setup cost. However, as a packet is transmitted to its destination, a routing decision has to be made at each node along the route. For a short communication session, such as transmission of a short text message, using datagram switching is a reasonable solution. A short text message can be transmitted by a few packets. Thus the total time spent on routing decisions by nodes along the route is small. On the other hand, multimedia communication sessions usually involve transmissions of larger numbers of packets. Then, datagram switching may consume a large amount

of time and computing power by the decision making processes. In such a situation, virtual circuit switching may be more practical.

In virtual circuit switching, a virtual circuit is established between the source station and the destination station. Such circuit is set up by a handshake process performed by two end stations. After the circuit is established, all packets of a given communication session are transmitted over it. The setup involves an initial cost, but stations along the virtual path do not need to make any routing decisions. Using virtual circuit switching for sending messages consisting of a larger number of packets is more efficient than using datagram switching. The cost of the initial setup phase in virtual circuit switching can easily be compensated for by savings in the route decision making process at all stations needed in datagram switching. The other advantage of using virtual circuit switching is the capability of providing Quality of Service (QoS) to different communication sessions. In virtual circuit switching, each communication session can be given one or more virtual circuit. As a result, different amount of bandwidth can be assigned to each communication according to its needs. Despite of these advantages, virtual circuit switching has not been advocated in ad–hoc networks. The reason for this is that most medium access control (MAC) protocols proposed in the past for ad–hoc networks would not be able to support "virtual circuits". This problem has been weakened by recently proposed protocols; therefore, as we will show, it is now possible to implement a traffic control scheme that uses virtual circuit switching in ad–hoc networks.

The main goal of the research reported in this thesis was to investigate the feasibility of applying virtual circuit switching in ad–hoc networks. A protocol that uses the virtual circuit switching technique, named Ad–Hoc Virtual Switching Routing, has been proposed as the outcome of our research. AVSR adopts the routing mechanism from Dynamic Source Routing (DSR) for obtaining route information and provides additional functions to set up a virtual path between a given the source and destination

station. We studied the performance of AVSR in a simulated network consisting of a number of mobile wireless stations. It was shown that it is applicable to implement virtual circuit switching in an ad–hoc network, however its performance degrades significantly as the number of nodes/terminals in the network increases. Such a result gives recommendations for future researches of virtual circuit switching in an ad–hoc network.

1.2 Thesis Layout

This thesis reports our study of virtual circuit switching in ad–hoc networks. Currently proposed traffic management schemes of ad–hoc networks, which include MAC and routing protocols, are shown in Chapter 2. Both datagram and virtual circuit switchings are discussed in Chapter 3. It also includes the discussion of possible implementation of virtual circuit switching in ad–hoc networks. A novel routing procedure for ad–hoc networks, called AVSR, is described in Chapter 4. Chapter 6 presents the results of performance evaluation of AVSR. The design of our simulation model used for evaluation is shown in Chapter 5. Chapter 6 presents the results of performance evaluation of AVSR. Chapter 7 concludes the thesis and discusses possible future works for this research.

Chapter 2

Routing and Medium Access Control Protocols in Ad Hoc Networks

This chapter presents the up–to–date survey of current medium access control (MAC) protocols and routing algorithms proposed for ad–hoc networks. A routing algorithm is a collection of instruction that describes how a network node finds the route(s) to another node. Routing algorithms for ad–hoc networks can be classified into three categories. They can be flat, hierarchical and geographic position assisted routing. Additionally, flat routing algorithms can be further divided into proactive and reactive routing.

A medium access control (MAC) protocol specifies how the medium is shared and how nodes access the medium. At present, *Carrier Sense Multiple Access with Collision Avoidance*(CSMA/CA) is the most popular MAC protocol assumed in research related to ad–hoc networks. It is also the fundamental technology for the popular IEEE 802.11x family. However, CSMA/MA is not perfect for ad–hoc networking. For example, it does not solve the hidden terminal problem completely (see Section 2.2.1).

This chapter is structured as following: an overview of routing algorithms in ad–hoc networks and their applications are firstly presented. Then follows the description of CSMA/CA in detail. Some other MAC protocols are briefly discussed before the chapter summary.

2.1 Routing in Ad–Hoc Networking

A routing algorithm describes processes that should be executed for obtaining route information between network nodes. Routing algorithms proposed for ad–hoc networks can be broadly categorised into three types: flat, hierarchical and geographic position assisted [9]. Flat routing algorithms assign every node in the network equal functionalities, while hierarchical routing algorithms usually assign different duties to different network nodes. Geographic position assisted routing algorithms use the Global Positioning System (GPS) to enhance decision making by providing the geographical information of network nodes. A description and examples of each category are presented in the following subsections.

2.1.1 Flat Routing Algorithms

Flat routing algorithms can be divided into two groups: proactive and reactive routings. The main common feature of proactive routing algorithms is that routing information is exchanged between network nodes. This happen whether or not the information is required for a communication session. The reactive routing algorithms, on the other hand, do not have any routing activities prior to the start of a communication session.

Proactive Routing

Proactive routing is also referred to as table–driven routing. It requires every network node to have one or more routing tables that provides up–to–date routing information

to all neighbours. Every node periodically broadcasts route update packets which are used for exchanging routing information updates. The main advantage of proactive routing is low–latency route access. However periodic updates can cause routing overheads in the network. Some of ad–hoc routing algorithms based on proactive routing are described in the following:

Destination Sequenced Distance Vector Routing (DSDV) DSDV [24] is a routing algorithm based on the classical distributed Bellman-Ford routing algorithm. Each node in the network maintains a routing table where all possible immediate destinations, the number of routing hops and the sequence number to each destination are stored. The sequence number is used for distinguishing stale routes from new routes. To minimise control traffic overheads caused by periodic updates, two types of route update packets are used in DSDV. A *Full dump* update carries all available routing information. This type of update can require multiple packets for transmission. The other type of update, *incremental*, is only responsible for updating recently changes of topology and it can always be done by sending one packet. A *full dump* update is used occasionally when a node detects a major network topology change. Between any two *full dump* updates, *incremental* updates are used.

Fisheye State Routing (FSR) FSR [22] is based on link state (LS) routing algorithm. It differs from the conventional LS algorithm in the ways that routing information is updated. Using the conventional LS algorithm, a link state update is exchanged when a link state change occurs. In contrast, nodes using FSR perform link state updates periodically with their neighbours, and the update frequency between two neighbours is determined by their distance apart. A node sends link state updates to a close neighbour more frequently than to a faraway neighbour. FSR uses this approach to minimise the update overheads in a large scale network.

Optimised Link State Routing (OLSR) OLSR [3] is another routing algorithm based on link state routing. This protocol uses *multipoint relaying* (MPR) [27] to minimise flooding traffic control packets. In a network using OLSR, nodes broadcast Hello messages frequently. An HELLO message contains a list of an initialler's immediate neighbours. By listening to HELLO messages sent by immediate neighbours, a node can figure out the nearby network topology within two hops distance. The node then computes the minimum set of one hop relay points required to cover all neighbours that are two hops away. Such a minimum set is called a MPR set. After a node computes its MPR set, it includes the MPR set into its HELLO messages. As a node receives such an HELLO message which has a MPR set, the node records the other nodes that mark it as their MPRs (called *MPR selector*). If a link state update is issued by node A, only its immediate neighbours that mark node A as their *MPR selector* will forward the update. Additionally, the link state update is reduced, because it only contains neighbours that select node A as their MPR.

Reactive Routing

Reactive routing is sometimes called source–initiated on–demand routing. Unlike proactive routing algorithms, reactive routing algorithms do not maintain up–to–date routing information on all nodes. When a node requires the route to a destination, it uses *route discovery process* to obtain the information. If this process is completed successfully, it returns the routing information to its initialiser, and updates routing information at all nodes along the route. Once the route is found, *route maintaining process* and *route recovery process* are used to ensure the validity of the route during the communication. Reactive routing reduces routing overheads by sending traffic control packets only when they are desired to a communication. On the other hand, it has long route access time because it has to wait for the reply from the *route discovery process*. Descriptions of some ad–hoc routing algorithms based on reactive routing is

listed in the following:

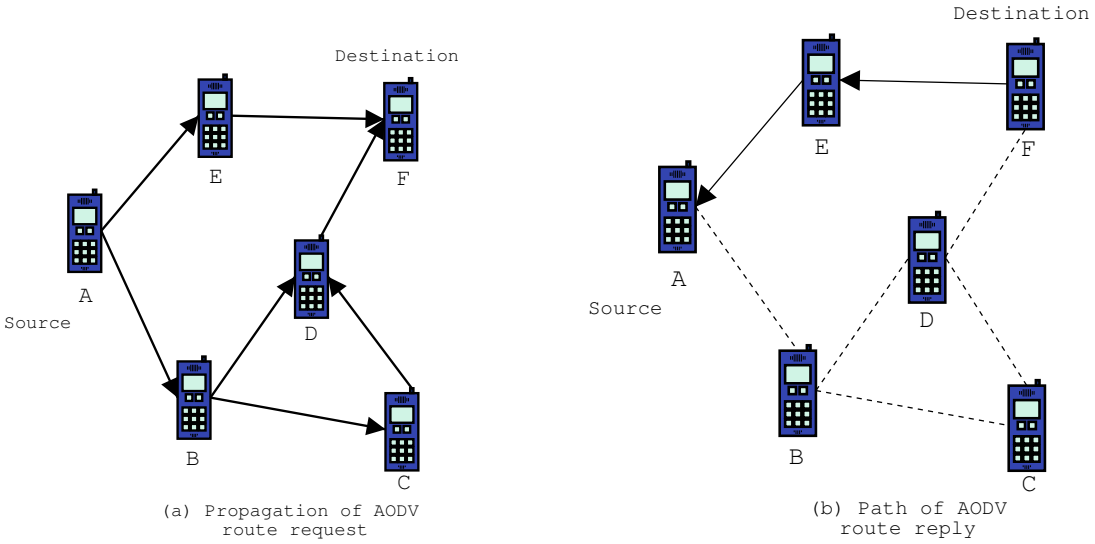


Figure 2.1: AODV *route request* and *route reply*.

Ad–Hoc On–Demand Distance Vector Routing (AODV) AODV [25] is built on DSDV, but it uses a reactive routing approach. Similar to DSDV, AODV uses sequence number to maintain the "freshness" of a route. When a source node wants to send a message to a destination but can not find a route to it, the source node initiates a *route discovery process* to locate the position of the destination node. The source node broadcasts a *route request* (REQ) which contains its address, sequence number, broadcast ID, the address of the destination and last known sequence number of the destination. The broadcast ID is increased by one after each REQ broadcasting in order to distinguish each REQ initialised by the same node. The combination of broadcast ID and the source's address identify the uniqueness of each REQ. A node receiving a REQ only rebroadcasts it if the node has not received a REQ from the same source address and with the same broadcast ID before. During the process of forwarding the REQ by intermediate nodes, a reversal route to the source node that

initiated the REQ is also setup. A REQ is propagated in the network until it reaches the destination node or a node which has the routing information to the destination node; however intermediate nodes can only reply to a REQ if they have a route to the destination with a destination sequence number than the destination sequence number in the REQ. When the destination node or a intermediate node replies a REQ, it sends a *route reply* (RREP) to the source node by using unicast[1]. As the RREP is sent back to the source node, intermediate nodes along the path setup a route entry to the destination node.

Routes in AODV are maintained by two methods. If the source moves and break the current route, it starts a new *route discovery process* to find a new route to the destination. If an intermediate node along a route detects a link failure, the other method is used. This method requires the node which detected the link breakage to send a *link failure notification* to the source node. After the *link failure notification* is received by the source node, the source node may start a new *route discovery process* if a route to the destination is still desired. An example of AODV *route request* and *route reply* are shown in Figure 2.1.

Dynamic Source Routing (DSR) DSR [11] is a routing algorithm based on source routing. Similar to AODV, the source node starts a *route discovery process*, when the source wants to send a message to a destination node but can not find the route there. The *route request* packet broadcasted by the source node contains the address of the source node, a broadcast ID, the address of the destination, and a route record. When a node receives a *route request*, it appends its address onto the route record within the request packet and forwards the packet to its neighbour; however, in order to limit *route requests* propagating in the network, the node should discard the request if its

[1]A packet sent by using unicast will only be received by a specific node and it will be ignore by other nodes that also receive the packet.

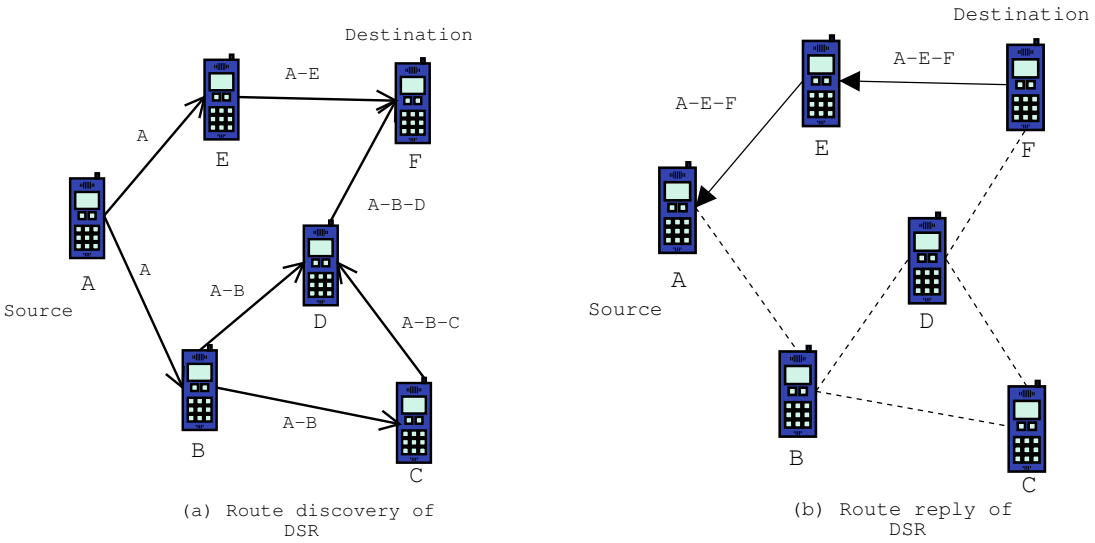


Figure 2.2: DSR *route request* and *route reply*.

address can be found in the route record. The *route request* is propagated in the network until it reaches the destination node or an intermediate node that has the route information to the destination. The destination node or the intermediate node only replies to the first received *route request*. Additionally, if the request is replied by an intermediate node, the intermediate node has to append its route associating with the destination onto the route record in the request; a complete route from the source to the destination node has to be included in the *route reply*. The node replying to a *route request* constructs a *route reply* and sends it to the source node by using its own route to the source node. If the replying node does not have the route to the source node, it can start a new *route discovery process* and append the reply onto the *route request*. The other method for obtaining a route to the source is to reverse the route record in the *route request*, but it is only possible if systematic link is possible in the network.

The route maintenance is achieved by receiving acknowledgements and *route error* messages. A node will send a *route error* if it detects a broken link for the route. Nodes

receiving the *route error* should give up that route, and start a new *route discovery process* if that route is still desired. Acknowledgement messages are used to verify the validness of a route. An acknowledgement can be sent explicitly by the destination of a route or as appendices in other types of traffic, such as TCP acknowledgement. An example of DSR *route request* and *route reply* are shown in Figure 2.2.

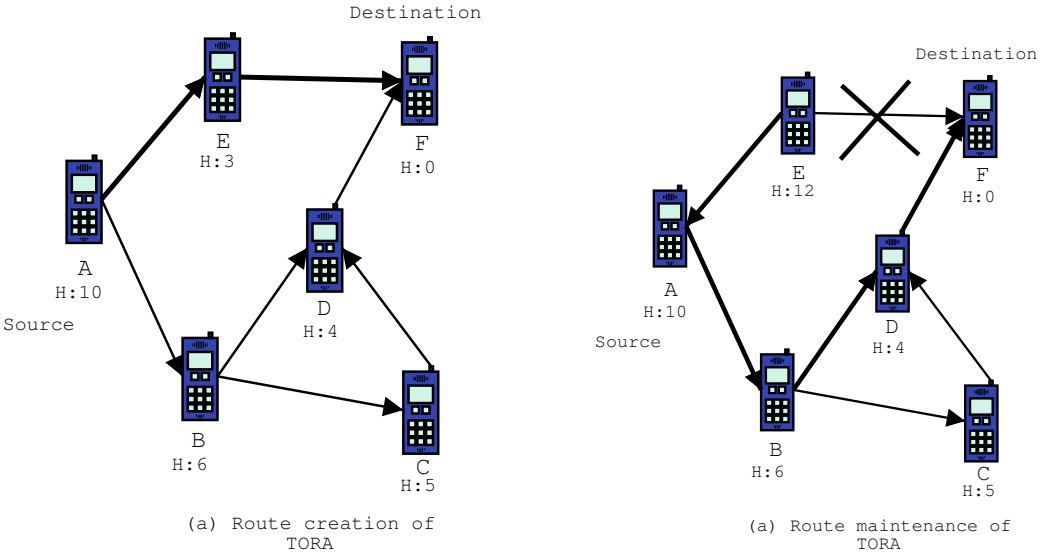


Figure 2.3: TORA *route creation* and *route maintenance*.

Temporally Ordered Routing Algorithm (TORA) TORA [19] is a high adaptive routing algorithm. The main key concept of TORA is to keep traffic control messages propagating to a small set of nodes that are near to the location of topological change. Similar to other reactive routing algorithms, it has three basic functions: *route creation*, *route maintenance* and *route recovery*. During the *route creation* and *route maintenance*, nodes use a "height" metric to establish a *directed acyclic graph* (DAG). The source node has the highest metric, and the destination node has the lowest. Intermediate nodes between the source and destination node have a metric based on their "transmission cost" to the destination node. The link direction between two

immediate nodes is determined by their relative "height". Traffic always travels from a node that has a higher "height" metric to a node having a lower metric. When selecting a route from the source node to the destination node, a link with higher relative "height" is preferred. The process of setting up a DAG is similar to the query/reply process proposed in Lightweight Mobile Routing (LMR)

The route maintenance is achieved by change the "height" metric of the node that detects a broken link. After a node detects it has a broken link, it resets its "height" metric to a value higher than the source node; thus traffic using it as an intermediate node will flow back to the source node. The source node then uses another route for sending traffic to the destination based on the new DAG. An example of *route creation* and *route maintenance* is shown in Figure 2.3.

2.1.2 Hierarchical Routing

The key idea of hierarchical routing is the decomposition a large network into smaller subnets. Nodes are assigned different functionalities inside and outside a subnet. The most popular way of building the hierarchy is based on the geographical location of nodes. Nodes that are close to each other are grouped as a cluster, and one node is selected as the cluster–head of this subnet. A node that is grouped in two or more clusters is used as the gateway. The advantage of hierarchical routing is its scalability. Generally current flat routing algorithms become inapplicable when the network size exceeds certain thresholds, because of link and processing overheads[9]. Decomposing a large network into smaller subnets and forming a hierarchy helps solving the scalability issue; however new overheads are introduced from clustering and assigning different functionalities to nodes. Examples of routing algorithms in this category are shown in the follow.

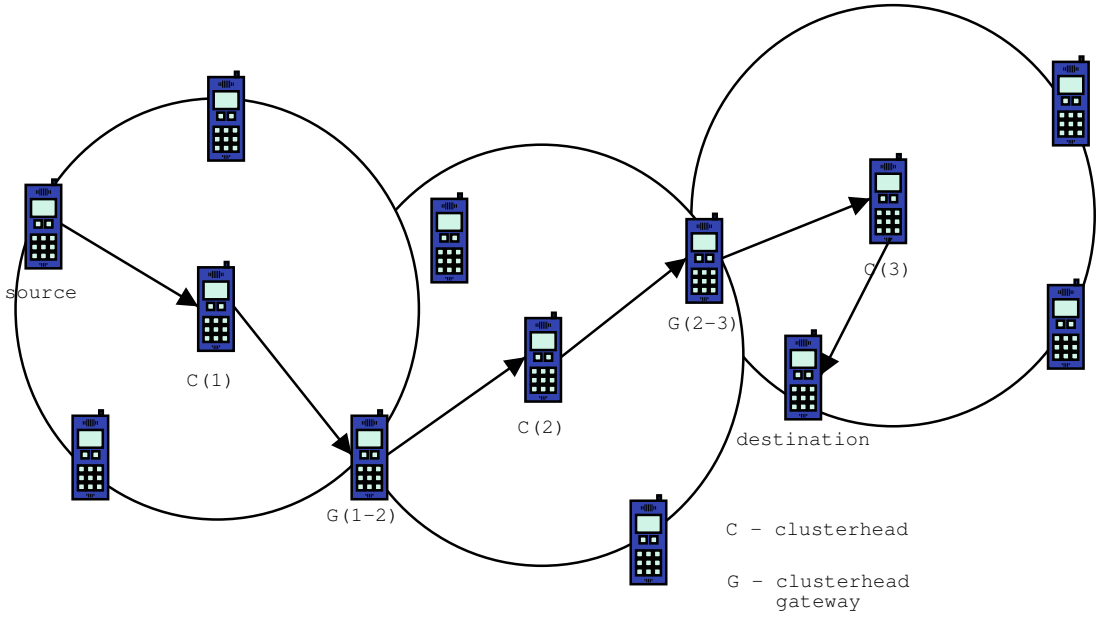


Figure 2.4: An example of CGSR routing.

Clusterhead–Gateway Switch Routing (CGSR) CGSR [2] uses *Least Clusterhead Change* (LCC) to partition the whole network into clusters. In each cluster, one node is elected to behave as the *cuslterhead*. Nodes existing in more than two clusters are used as *cluster gateways* between these clusters. The distance between a *clusterhead* and its cluster members is normally one hop. The *clusterhead* has two tables to store routing information. One table is used for storing its member nodes where the *clusterhead* receives periodical updates. The other table stores the gateway for routing packets to the *clusterhead* of nodes in other clusters. When a packet is sent by a node, it is firstly transmitted to the *clusterhead* of the current cluster. If the destination node of this packet is in the same cluster, the *clusterhead* then forwards the packet to the destination node. Otherwise, the *clusterhead* routes the packet to another *clusterhead* via a gateway, and so on until the packet reaches the *clusterhead* of the destination node. CGSR can greatly reduce the routing table size by having only one entry for all nodes in the same cluster; however, new overheads are intro-

duced from the set up of clusters. Additionally, if clusters changes frequently, the performance of CGSR can be affected negatively. An example of CGSR routing is shown is Figure 2.4. In this example the source node sends a packet via the route: source→C(1)→G(1–2)→C(2)→G(2–3)→C(3)→destination. This route is clearly not the shortest path because *cluster gateway*G(2–3) can forward traffic to the destination directly without using the *clusterhead*C(3) as a relay; however, a *cluster gateway* can only communicate with a *clusterhead* but no other immediate neighbours in CGSR. Thus, shortest path between the source and destination nodes is not guaranteed in CGSR.

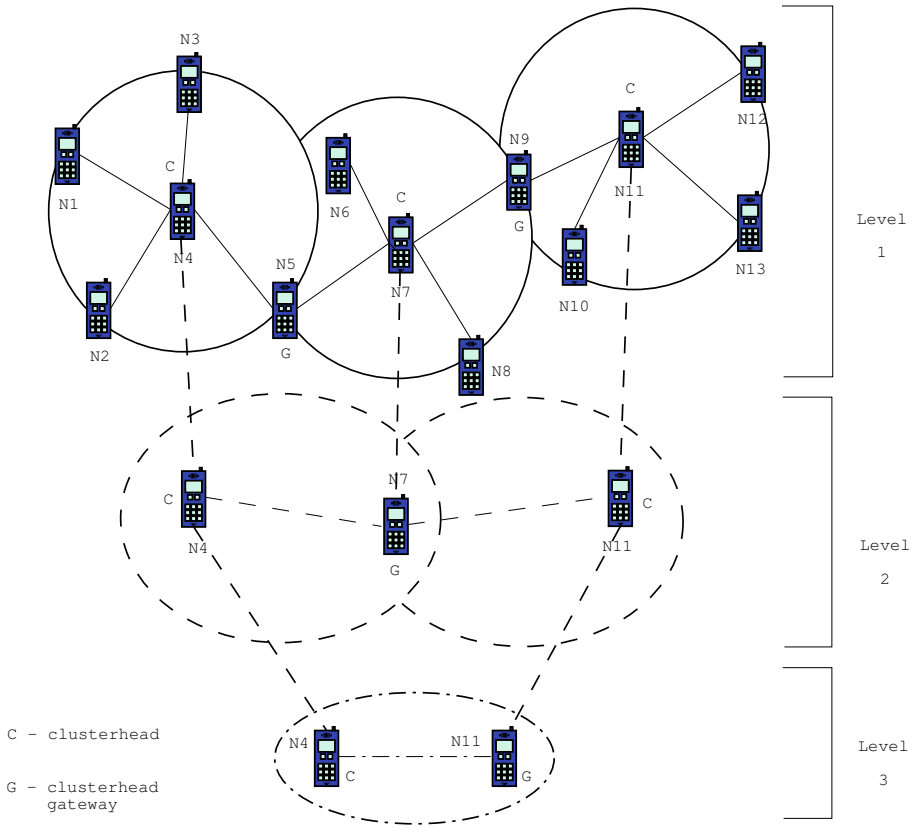


Figure 2.5: An example of HSR multilevel clustering.

Hierarchical State Routing (HSR) HSR [23] is based on the link state routing algorithm with multilevel clustering. A logical hierarchical topology is maintained by applying a clustering scheme recursively. Nodes at the same logical level are grouped into clusters, and those elected *clusterheads* at the current logical level become members at the next higher logical level. An example of multilevel in HSR is shown in Figure 2.5. A *hierarchical ID* (HID) is used to represent a node in HSR. A HID is defined as a sequence of physical addresses of the nodes on the path from the highest hierarchy to the node itself. For example, in Figure 2.5, the HID of node 6 (N6) is <4,7,6>. By examining the HID in a packet, that packet can be routed to any destination in the network topology. When a node does not have enough information to route a packet, it then forward the packet to another at one higher level and so on. After that packet reaches a node at a higher level, which has the information to route the packet, it is then forwarded to its destination. For example, as shown in Figure 2.5, node 1 (N1) sends a packet to node 6 (N6). The packet has a HID <4,7,6>. N1 does not have the route information to N6 itself, so it forwards the packet to its upper hierarchy, node 4 (N4). N4 still does not have the route information to N6, but the level 2 address of the HID in the packet is node 7, which is a member of N4 at level 2. Thus, N4 forwards the packet to N7 based on its level 2 routing table. After the packet reaches N7, N7 then forwards the packet to its member node N6. In HSR, nodes can update their route information dynamically and locally on receiving updates from nodes at a higher hierarchy. However, this benefit incurs overheads from forming clusters and having longer HID addresses. Similar to CGSR, HSR also has performance degradation issues as the formation of clusters changes rapidly.

Zone Routing Protocol (ZRP) ZRP [8] takes advantages from both proactive and reactive routings. There are two layers of hierarchies in ZRP: nodes inside a zone and nodes outside a zone. The zone size is determined by the number of hops

and is predefined in the network. Any proactive routing algorithm can be used for routing packets to destination nodes which are inside the zone; for routing packets to destination nodes outside the route, any reactive route can be used. ZRP has three components: *Intrazone Routing Protocol* (IARP), *Interzone Routing Protocol* (IERP) and *Route Query/Route Reply*(RREQ/RREP). IARP maintains up–to–date routing information within the zone, and IERP uses RREQ/RREP to discover route information outside the zone when they are desired. ZRP tries to minimise periodical updates in a limited zone, and route discovery overheads to only selected border nodes. However, it is based on the assumption that nodes are more likely to communicate with neighbours which are closer to themselves. If this assumption fails, ZRP will behave as a reactive routing algorithm.

2.1.3 Geographic Position Assisted Routing

With the assistance from a Global Position System (GPS) device, a wireless node is capable of obtaining its geographic location precisely. After nodes exchange their geographic location information, they are able to build a clear network topology map. This map is then used to improve routing efficiency. A geographic position assisted routing algorithm is usually based on one of flat or hierarchical routing algorithms and uses geographical information to improve its performance. Additionally, GPS can also provide universal timing, and assists global time synchronisation in an ad–hoc network. Description of some geographic position assisted routing algorithms are presented in the following:

Geographic Addressing and Routing (GeoCast) GeoCast [17] allows a node to send messages to all nodes in a specific geographical region. It is achieved by using a geographical address which can be a point, a circle described by a point and its radius, or a polygon described by a list of points. Instead of using networking addresses, a geo-

graphical router (GeoRouter) uses geographic addresses to route packets. GeoRouters in the network negotiate with each other and form a hierarchical structure. When a GeoRouter does not have enough information to route a packet, it will forward the packet to another GeoRouter a one higher level. This routing process is similar to the hierarchical routing approach. The main advantage of GeoCast is multicasting or group reception; however a geographical address can be long, and setting up a hierarchical structure for GeoRouters can be complex.

Location Aided Routing (LAR) LAR [12] is based on a reactive routing algorithm, DSR, and uses geographic information to control the direction of traffic control overheads which are being flooding in the system. According to the previous known location of the destination node, the sender transmits the route request packet to an expected region where the destination node is. This expected region is attached to the route request packets. During the route request propagation to the destination, only nodes inside this region can forward the request. There is another approach which uses the distance between the source and destination nodes to control the route request packet. Before the source initiates the route request packet, it calculates the distance between itself and the destination node. The distance is then included in the route request packet. When a node receives such a request, it will only relay the packet if the distance between the current node and the destination node is shorter or equal to the distance included in the route request packet. Although LAR can limit traffic control overhead caused by route request flooding, the decision making process on a node is more complex than the original DSR.

2.2 Medium Access Control in Ad–Hoc Networking

A medium access control (MAC) is a set of procedure that describes how a terminal accesses the medium. One major functionality of a MAC protocol is minimising the probability of having a collision. When more than one terminal accesses the medium at the same time, their signals will be interfered with or even destroyed by each other; thus the receivers are not able to decode the incoming signal successfully. A MAC protocol that provides low collision probability is essential for good performance networking.

2.2.1 Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA)

In a wireless network with central administration, the network can use a centrally controlled MAC protocol; the wireless medium is reserved by the central controllers at all time. In such a system, no terminal can transmit data until it has been granted medium accessing permission from a central controller. Therefore, as long as the central controllers do not assign the same physical medium to more than one terminal at the same time, collisions can be avoided.

An ad–hoc network does not have fixed network nodes to behave as central controllers, because each node has the same mobility in the network. Thus, a centralised MAC control is not suitable in an ad–hoc network. At present, the most popular MAC protocol for ad–hoc networking is Carrier Sense Multiple Access (CSMA). This is because of its simple and well understood design. CSMA is a MAC protocol that uses single frequency band and can function in a distributed manner. The basic principle of CSMA is “listen before start talking”. In a network uses CSMA, a node senses if there is already data transmitted in the medium before the node accesses the medium. The

node has to abort its medium access attempt if it senses data in it. CSMA can avoid collision sufficiently if the sending terminal is within the sense range of the current node that is willing to access the medium. However, collision can still occur in CSMA.

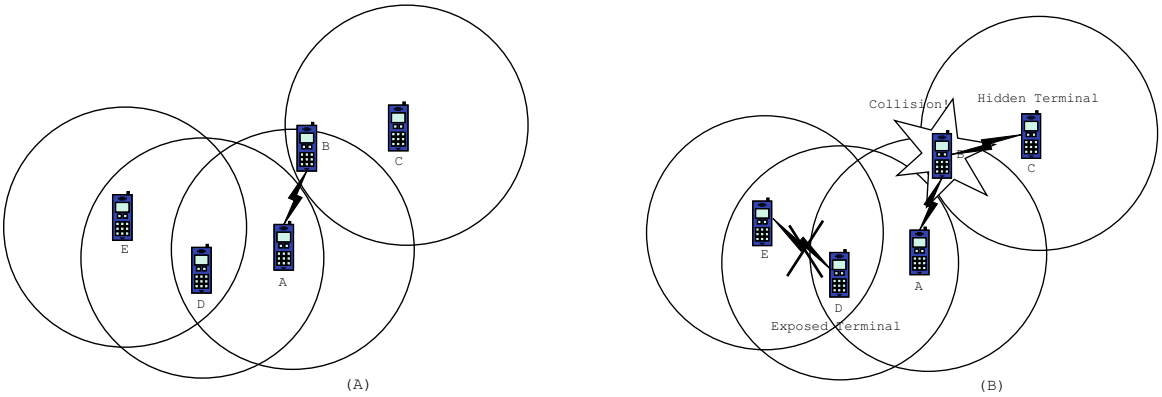


Figure 2.6: The hidden and exposed terminal problem.

An example of a small network is shown is Figure 2.6. In this example, node A and C are out of signal range (represented by the large circle) to each other. Therefore, node C can not sense the communication between node A and node B. When Node C starts its transmission to Node B, it then causes a collision. In such a situation, node C is called as a hidden terminal and should be limited form accessing the medium. In contrast, node D senses the transmission of node A, and is restricted for accessing the medium. However the transmission range of node D does not cover node B; thus its transmission to node E does not interfere the communication session between node A and node B. Forbidding node D from access the medium then make it become an exposed terminal in the system.

To solve the hidden and exposed terminal problem in CSMA, a collision avoidance scheme (RTS–CTS) is introduced. This solution is also refereed as Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA).

RTS–CTS RTS (Request To Send) and CTS (Clear To Send) are control messages for reserving the wireless medium. RTS is broadcasted by the terminal that wishes to transmit data. The destination terminal then broadcasts a CTS message, which informs the requester to start its transmission. During this handshake process, all other terminals who received both a RTS and a CTS message are blocked from accessing the medium until further notice or a medium reservation timeout. Terminals who receive a RTS without a CTS are able to start new RTS–CTS handshakes, but can not acknowledge a received RTS. In contrast, terminals who receive a CTS without a RTS are able to acknowledge a received RTS, but can not start new RTS–CTS handshakes. An example of a RTS–CTS handshake is shown in Figure 2.7. In this example, node A wishes to transmit a message to node B. To reserve the wireless medium, node A broadcasts a RTS message. This RTS message blocks node D from accessing the medium and notifies node B about the request. Node B then broadcasts a CTS to confirm the request from Node A. This CTS message also blocks Node C from starting a RTS–CTS handshake. After Node A receives the CTS, it start transmitting data to Node B. When the transmission is over, Node B broadcasts an acknowledgement (ACK) message to Node A. The ACK message also notify Node B about the end of medium reservation. In this example, Node D does not receive a CTS message after receives a RTS message. Therefore, it can start a RTS–CTS handshake to node E while node A is transmitting data to node B. Similarly, node C is allowed to start a new RTS–CTS handshake because it only receives a CTS message.

RTS–CTS scheme can minimise the probability of having collision during a transmission session, but it has some limits. It can not reduce the probability of collision until RTS–CTS handshake is completed successfully. Thus, the probability of having a collision during RTS–CTS handshake is relative higher. Additionally, in certain situation, as shown in Figure 2.8, collisions can also happen to data packets because of incomplete RTS–CTS handshake. This case is caused by the limit of wireless com-

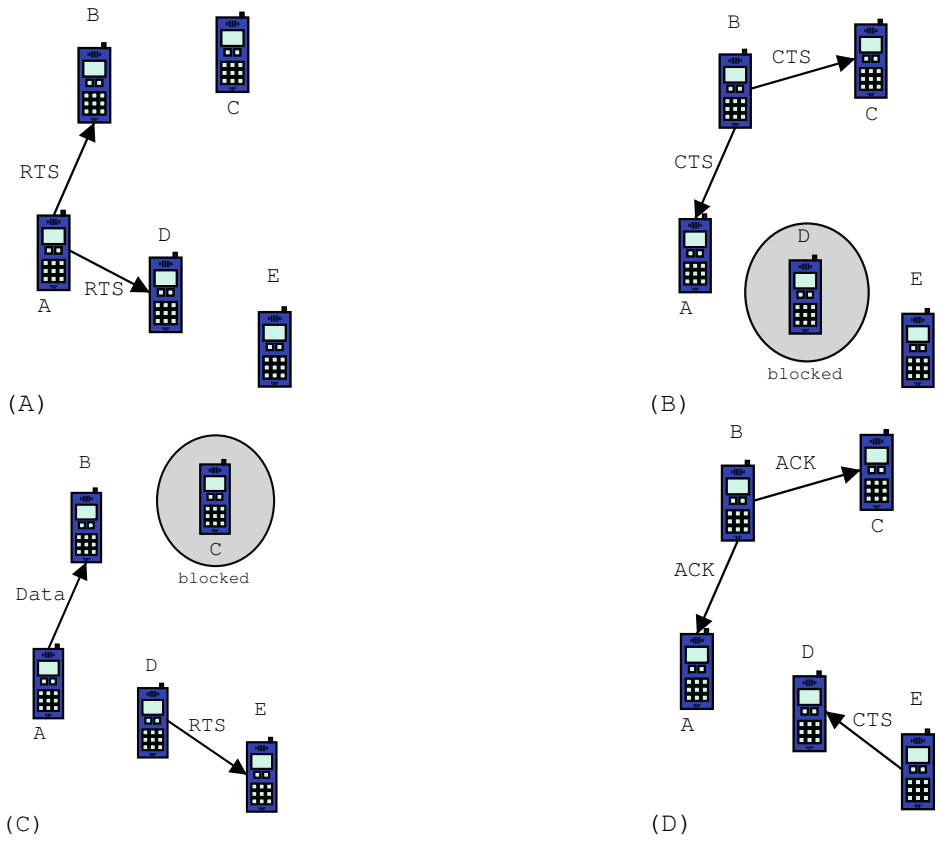


Figure 2.7: An example of RTS–CTS–DATA–ACK handshake.

munication that a node can not perform transmitting and receiving data at the same time.

An additional advantage of using RTS–CTS messages is power control. When a node is blocked by a CTS message, the node knows it will be a short while before it can access the medium again. Thus, the node can switch off for power saving in its idle period. It is an attractive feature for ad–hoc networks, because most wireless devices in ad–hoc networks have limited and finite power.

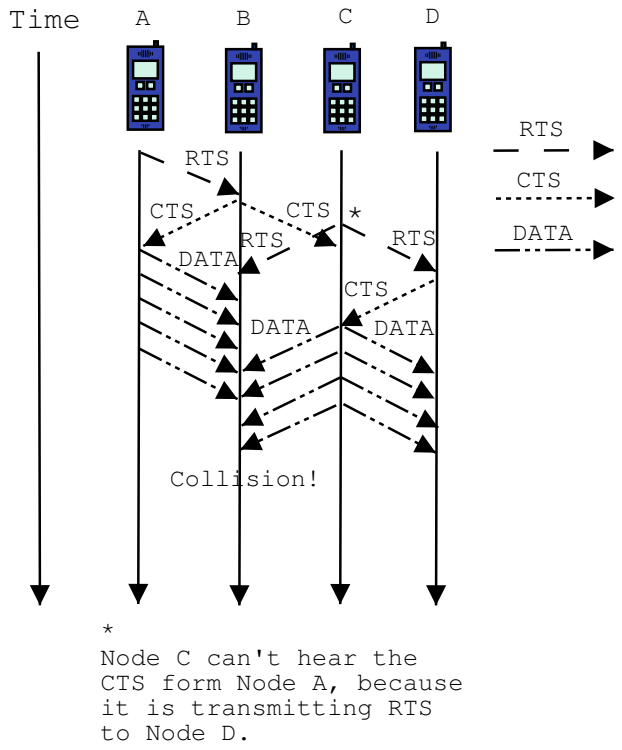


Figure 2.8: An example of incompleting RTS–CTS handshake, which causes collisions.

2.2.2 Other MAC protocols

At present, CSMA/CA is the most popular MAC protocol used in research [1] [13] [26] that is related to ad–hoc networks. However, there are also some other options available[2], which were rarely used in previous research related to ad–hoc networks. These MAC protocols are briefly discussed in the following:

Multiple Access with Collision Avoidance (MACA) MACA was designed for creating a single–frequency and usable ad–hoc network. RTS–CTS–DATA handshake is used in MACA to avoid collisions. Unlike CSMA, a node does not perform carrier sensing before transmitting data. A node can send a RTS message at any time; therefore, collision probability is higher during the RTS–CTS handshake in MACA. If more

[2]see Chapter 4, Ad Hoc Mobile Wireless Networks:protocol and system[29]

than one node transmit a RTS message concurrently, a collision can occur. In such a situation, nodes that were transmitting RTS messages will wait a random amount of time before retransmitting their RTS messages.

Multiple Access with Collision Avoidance–By Invitation (MACA–BI) Unlike MACA, there is no RTS message in MACA–BI, and the CTS message is renamed as RTR (Ready To Receive). In MACA–BI, a sending node can not transmit data unless it has received an invitation from the receiving node. However, the receiving node does not really know if the sending node has data ready for transmitting. The receiving node has to predict the intention of the sending node based on traffic arrival rate and other parameters. The configuration of these parameters are closely correlated to the network performance.

Dual Busy Tone Multiple Access (DBTMA) DBTMA uses two out–of–band busy tone channels: Transmit Busy Tone and Receive Busy Tone, to notify neighbours of all on–going transmission. These two busy type tones are carried by two different frequency spectrum; thus they do not interfere each other. When a node wants to transmit, it sends a RTS message to the receiving node. If the receiving node is available for receiving data, it replies with a CTS message to the sending node, and starts transmitting busy tone in the Receive Busy Tone channel. After the sending node receives the CTS message, it starts transmitting data and busy tone in the Transmit Busy Tone channel. Nodes receiving a busy signal from either of the busy tone channels are prohibited from accessing the medium.

2.3 Summary

In this chapter, we presented an overview of routing algorithms and MAC protocols in ad–hoc networks. Our literature survey shows that each category has its strengths

and weaknesses, and no particular routing algorithm is ideal for all scenarios.

CSMA/CA is the most popular MAC protocol used for ad–hoc networking at present; however it can not solve the hidden terminal problem completely. Our study also indicates that CSMA/CA is not capable for supporting virtual circuit switching. Thus an alternative MAC protocol is desired in order to implement virtual circuit switching in an ad–hoc network.

The routing algorithm helps a wireless terminal to find the route(s) to another node; the MAC protocol instructs how the terminal access the medium. However, to setup a completed communication session, one more element is required, a strategy that describes how intermediate nodes along the route process the receiving messages. This strategy is refereed as switching technique, and is discussed in the next chapter.

Chapter 3

Virtual Circuit Switching Concept in Ad–Hoc Networks

Virtual circuit switching is a process of establishing a temporal connection between two logical links for a duration of a communication session. This operation allows intermediate nodes to set up a temporary path between two nodes, which are communicating with each other. If these two logical links are reserved during a communication session, a virtual circuit dedicated to this particular communication session is established. This approach enables a network to provide a more predictable and stable service because the quality of the communicating channel is obtained prior to the start of the communication session. However virtual circuit switching has rarely been used technique in ad–hoc networks because of such difficulties as time synchronisation and frequency management. Recent research [21] [18] has resolved these issues; thus we are now able to investigate the behaviour of virtual circuit switching in an ad–hoc network.

This chapter first discusses two switching techniques used in packet–switching networks, and follows the investigation of virtual circuit switching in ad–hoc networks.

3.1 Switching Techniques

To set up a communication session successfully, we need a MAC protocol for accessing the medium, a routing algorithm for obtaining route information from the source to the destination and a switching technique for establishing a temporal connection between them. The description of MAC protocols and routing algorithms can be found in Chapter 2.

A switching technique is a method that specifies the type of connection between two nodes and how intermediate nodes along the path manage traffic flows. There are two switching techniques available and they are referred to as *datagram switching* and *virtual circuit switch*[28]. The main difference between these two techniques is the way that an intermediate node handles receiving packets. An intermediate node treats each received packet independently in a network using datagram switching. In virtual circuit switching, an intermediate node processes each received packets based on pre–configured settings. Further discussion of each switching technique is given in the following.

Datagram Switching In the datagram approach, each packet is treated independently, and previous routing decisions are not considered for the current decision making. It is possible for an intermediate node to route packets having the same destination through different routes. This could cause packets to arrive at the destination out of their original order. If the sequence of arrival packets is important, such as transmitting voice or video data, the destination node has to buffer its incoming packets, and sort them based on the sequence information in each packet.

Virtual Circuit Switching In a packet–switching network using virtual circuit switching, a particular link must be established from a given source to the destination before any packet is sent. To set up a preplanned link for a communication session,

a call setup phase is required, and it causes a delay before a node can send the first packet. In virtual circuit switching, packets can only travel following a single path. This technique ensures that the arrival packets are received in the same order as their original one.

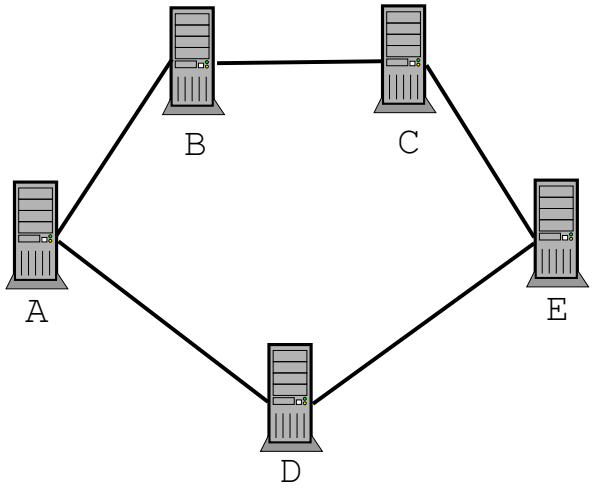


Figure 3.1: An example of a small packet–switching network.

Performance Comparison Figure 3.1 serves as an example for performance comparison. In such a network, if node A wants to send a message to node E, it has two possible routes: A→B→C→E and A→D→E. The following discussion shows how these two switching techniques handle this situation.

Datagram approach: A network using datagram switching makes routing decisions for each packet. Therefore it is possible that packets belonging to the same message are delivered via different routes and packets arrive at the destination out of their original order. An example of an event schedule during a transmission of an eight packet long message is shown in Figure 3.2. In this example, there is a communication session between node A and E. Packets belonging to this session are sent through two

possible routes: A→B→C→E and A→D→E. These two routes are used evenly in this example. One can clear see that packets do not arrive at node E in their original order because packets travelling via A→B→C→E have a longer transmitting time than packets travelling via the other route.

Datagram switching does not have any delay caused by an initial setup, but it takes more processing time for every node along the route. In addition, using this approach makes error control more difficult. For example, as shown in Figure 3.2, when node E receives the packet P2, it can not detect if the packet P1 has been lost or not. To perform the error detecting, node E has to buffer a certain amount of packets, sort them by their sequence number, and then determine if there are packets missing. Thus, early error detection is not possible for datagram switching.

Virtual circuit switching: An event schedule showing an eight packet long message being sent is shown in Figure 3.3. It shows the initial delay caused by the call setup process, and it also shows the main characteristic of virtual circuit switching, i.e. all packets travel in the same pre–organised route.

Although there is a setup overhead, the existence of a preplanned route saves the processing time at every node on the route. For sending a short message containing only a few packets, virtual circuit switching is less efficient than datagram switching because of its initial delay. However, virtual circuit switching can show its advantage for transmitting a long series of packets. Virtual circuit switching has a shorter processing time at every node on the route; therefore, the longer the series of packets are transmitted, the more time can be saved by using virtual circuit switching. Another important advantage is that packets arrive in order. Thus, if a packet arrives at a node without receiving a packet having the previous sequence number, an error is detected.

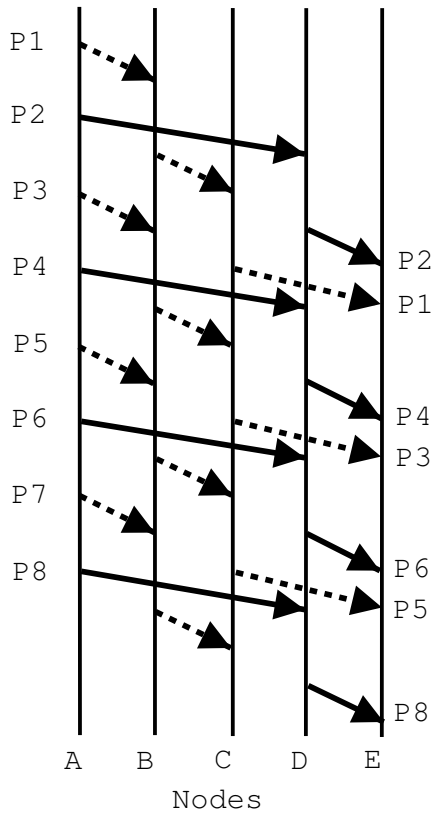


Figure 3.2: An even schedule diagram for sending a message from node A to node E by datagram switching.

3.2 Virtual Circuit Switching in Ad–Hoc Networks

During our investigation of how to introducing virtual circuit switching in ad–hoc networks, we looked at it from tree different aspects:

1. Issues of implementing a MAC protocol supporting the concept of "virtual circuit" in an ad–hoc network;

2. Interaction between different types of routing algorithms and virtual circuit switching;

3. The impact of rapid network topology changes to virtual circuit switching.

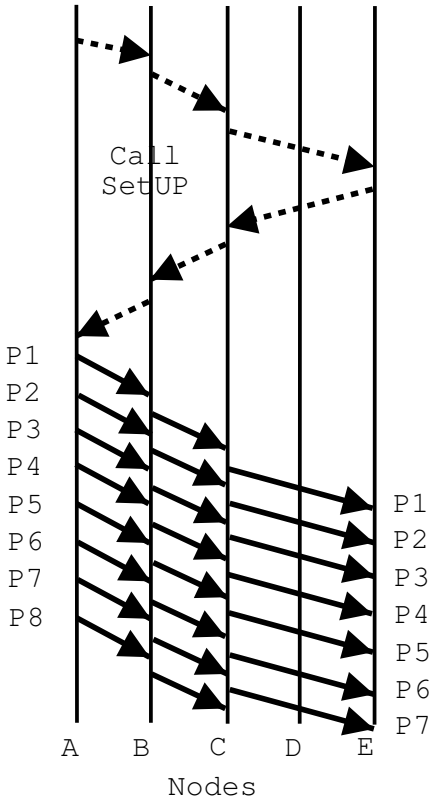


Figure 3.3: An even schedule diagram for sending a message from node A to node E by datagram switching.

3.2.1 Implementing A MAC Protocol Supporting Virtual Circuit

One important issue of using virtual circuit switching in ad–hoc networks is finding a medium access control protocol that supports the concept of virtual circuits. To use this concept, the medium must be accessed by either a time division or a frequency division multiplexing approach; thus there are "virtual circuits" in the medium. Currently, the most commonly used MAC scheme is Carrier Sense Multiple Access (CSMA), as discussed in Chapter 2, and it has no means for supporting the concept of virtual circuit. Thus, datagram switching has always been used in ad–hoc networks.

To use virtual circuit switching, a MAC protocol needs to have logical channels which act as "circuits". This can be achieved for accessing wireless medium by using frequency or time division multiple access. However, implementing a MAC protocol using frequency or time division in an ad–hoc network had been a challenge until recently. The lack of central administration made it difficult for an ad–hoc network to perform frequency management and time synchronisation. However, recent advancements of telecommunication research has changed this situation.

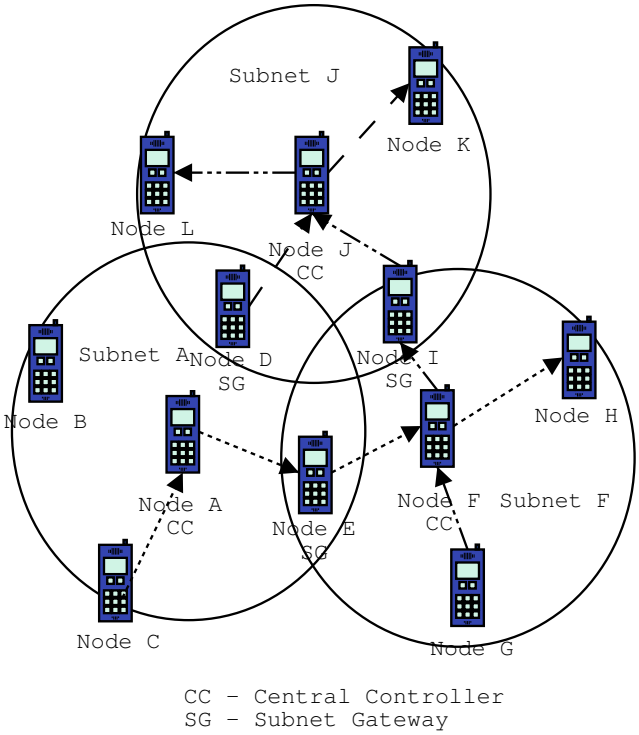


Figure 3.4: A HiperLAN/2 multihop ad–hoc network uses interconnection of subnets.

The major issue with using Frequency Division Multiple Access (FDMA) in an ad–hoc network is the need of central administration for frequency management. One possible method to solve this issue is selecting temporal central controllers in the network, and these central controllers are responsible for frequency management. Each central controller (CC) and its immediate neighbours forms a subnet. Subnets are

interconnected by terminals, called subnet gateways (SG), which associate with more than one subnet. Different frequencies are used in each subnet, and a subnet gateway changes its participating subnet by switching its transmitting frequency. This approach is called multiple–frequency forwarding [21]. An example of an HiperLAN/2 multihop ad–hoc network using this approach is shown in Figure 3.4.

The fundamental requirement for using a time division is time synchronisation for all nodes in the network. One method of fulfilling this requirement is using a GPS[1] receiver for obtaining time information from a GPS satellite. This approach is not suitable in some special situations, such as underwater or in space. In addition, a GPS receiver might be too large for small devices and also too expensive. Other time synchronisation techniques usually use neighbours' sending signal as the time reference, see as [18]. This techniques could introduce overheads into the network.

3.2.2 Interaction between Routing Algorithms and Virtual Circuit Switching

In Chapter 2, routing algorithms are classified into three categories: flat, hierarchical and geographic position assisted routing. However, we only investigated how virtual circuit switching interacts with flat routing algorithms. The other two categories were omitted because of their complexities. Because virtual circuit switching is not a well understood technique in ad–hoc networks, we tried to keep our investigation as simple as possible.

Flat routing algorithms can be further divided into two types, as described in Chapter 2. They are named proactive and reactive routing. The impact of the interaction between both types of flat routing and both switching techniques is discussed in the follwing:

[1] Global Position System

Using virtual circuit switching with a proactive routing algorithm, as shown in Figure 3.5(A), does not change either of their strengths or weaknesses. Virtual circuit switching still has its initial delay issue, and proactive routing has overheads caused by periodical updates. However, using virtual circuit switching with a reactive routing algorithm becomes a challenge.

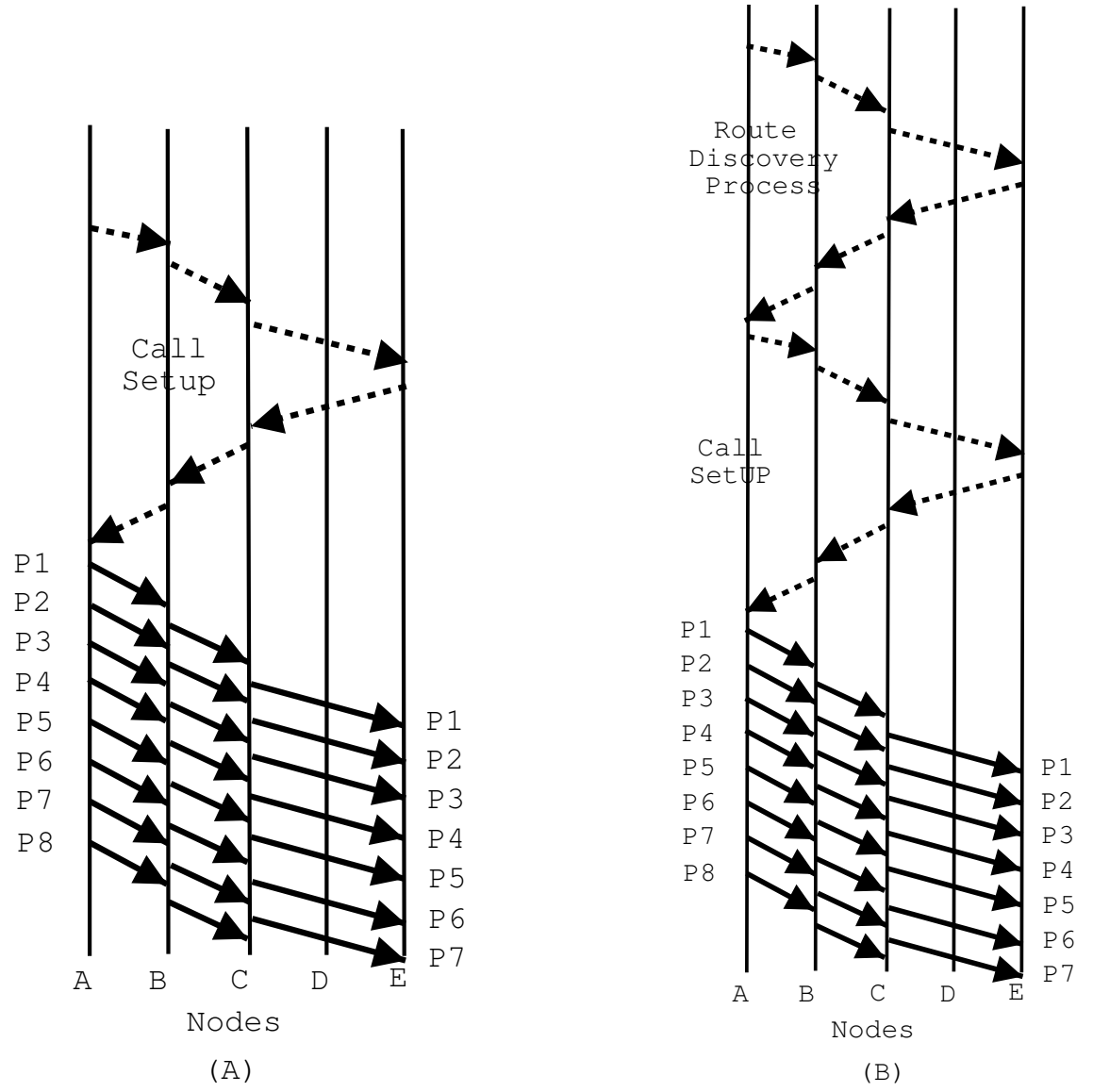


Figure 3.5: Virtual Circuit Switching interacts with two types of routing algorithms. (A):Proactive; (B):Reactive.

Virtual circuit switching and reactive routing algorithms both have initial setup overheads. In a system that uses virtual circuit switching with a reactive routing algorithm, it could have a long initial delay. Such a delay happens when two initialisations are issued sequentially, as shown in Figure 3.5(B). To improve such a situation, we see

an opportunity for integrating their initialisations, because they have a similar travel path, source→destination→source. If these two overhead processes are integrated into one, it could reduce the setup cost for both virtual circuit switching and the reactive routing algorithm very efficiently and effectively.

3.2.3 The Impact of Rapid Network Change

A rapid changed network does not create a friendly environment for virtual circuit switching, because it could frequently break up the virtual circuits. It costs additional traffic and time to recover a broken virtual circuit, and degrades network performance. Unfortunately, rapid network topology change is typical behavior for ad–hoc networks. Such a behaviour not only challenges the performance of virtual circuit switching, but also routing information maintenance. When the network topology changes, routing information on every node needs to be updated too. Proactive routing algorithms use periodic broadcasts to update the routing table on every node, and reactive routing protocols use route recovery process to reestablish a route after it is detected as broken. Here, we see another opportunity for integrating processes from virtual circuit switching and reactive routing algorithms, because the breakage of a route also causes a change of the virtual circuit. If there was a single process for recovering both the route and virtual circuit, it could minimise the impact of network topology change.

Another advantage of using virtual circuit switching is that it helps to minimise packet collision probability. During the call setup process, nodes are able to assign a time slot or frequency channel that is not being used by its immediate neighbours; thus the probability of receiving more than one packet at the same time at the same physical medium is reduced.

3.3 Summary

Two switching techniques, datagram and virtual circuit switching, were discussed in this chapter. In datagram switching, packets are treated independently and routing decisions have to been made at each intermediate node. A route is established between two nodes before data is sent when virtual circuit switching is applied. Both techniques have their own strengths and weaknesses. In general, datagram switching is a better choice for sending a short message. For sending a long message or having a long period of conversation, virtual circuit switching would perform more efficiently than datagram switching.

Datagram switching is a popular switching technique used in ad–hoc networks; however virtual circuit switching is not because of the lack of "virtual circuit" friendly MAC protocols. A "virtual circuit" friendly MAC protocol must use either a frequency or a time division approach. Recent advancements in telecommunication research have solved this issue; thus we are now able to investigate the behaviour of virtual circuit switching in an ad–hoc network.

During our investigation, we found the opportunity for merging control overheads in virtual circuit switching and reactive routing algorithms. This finding was also used in the protocol we designed for demonstrating virtual circuit switching in an ad–hoc network. The design of this protocol is presented in the next chapter.

Chapter 4

Design of the Ad–Hoc Virtual Switching Routing Protocol

Ad–Hoc Virtual Switching Routing (AVSR) protocol is a layer 2 switching control proposed in this thesis, and it covers the specification of a MAC protocol and a routing algorithm. AVSR is a traffic management protocol specifically developed for ad–hoc networking. This protocol demonstrates how we can use virtual circuit switching as a linking mechanism in ad–hoc networks and also demonstrates the probability of integrating this switching technique with a reactive routing algorithm. The integration is achieved by embedding the call setup in the route discovery process. This approach enables AVSR to minimise traffic control overheads and to shorten the initial delay. The general concept of AVSR has already been introduced can be found in Chapter 3. Our aim of developing AVSR is to investigate the application of virtual circuit switching in ad–hoc networks.

In this chapter, the protocol requirements are firstly discussed and analysed. Then, the design process of AVSR is presented. A small protocol demonstration is given at the end of this chapter.

4.1 Protocol Requirements

AVSR is a traffic control protocol designed for ad–hoc networks. As such, it has to satisfy some basic requirements specific for ad–hoc networks. These requirements are as follows:

- One of the main characteristics in ad–hoc networks is the lack of centralised controls. The absence of any infrastructure and the mobility of each node makes introduction of fixed administrative controllers difficult. Any traffic control protocol designed for ad–hoc networking should be able to function without infrastructure and without a central controller.

- The protocol must be able to detect any unexpected and unpredictable network failures and perform appropriate recoveries. These failures can be caused by sudden changes in a network's topology, including disappearance of nodes from the network.

- Wireless devices are usually powered by batteries. An ad–hoc traffic control protocol should be aware of the need of power–saving. Therefore, overheads caused by traffic administration should be minimised.

- Computing power of small wireless devices is limited. Complex computing algorithms, such as complex routing algorithms, should be avoided when implementing an ad–hoc traffic control protocol.

- The traffic control protocol should not use a routing algorithm that causes any deadlock or loops traffic flows in the network. It is important to detect undeliverable traffic flows and remove them from the network.

In additional to the above requirements, our protocol also has to use virtual circuit switching technique and a reactive routing algorithm in order to demonstrate the

concept we proposed.

4.2 AVSR Protocol Feature

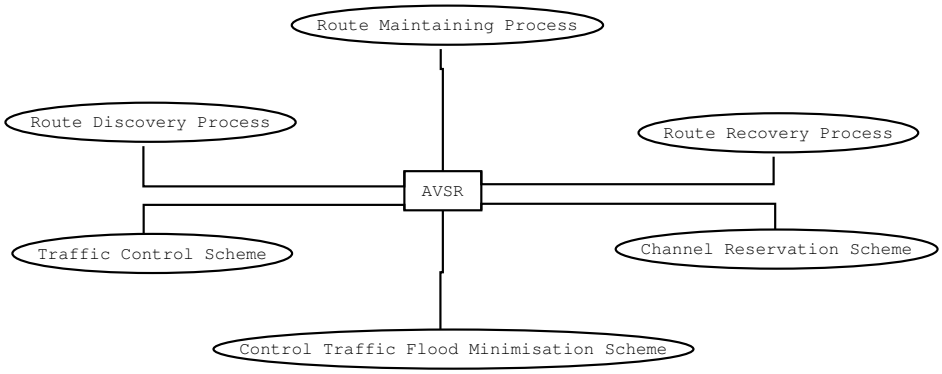


Figure 4.1: Main features of AVSR.

The main features that AVSR must encompass are shown in Figure 4.1. The discussion of each feature is given in the following.

Route Discovery Process: This process helps a source to find out the route to a destination node in the network. At eh end of this process an acknowledgement which returns the result of route discovery, is expected by the request initialiser. During the whole process, no central control should be needed; thus it should be a purely distributed process.

Route Maintaining Process: A node sending data traffic uses route maintaining process(es) to ensure the validity of the current active route(s). A route maintaining process is responsible for detecting and reporting the changes in the network topology to the node that owns the current process.

Route Recovery Process: After an active route is reported as broken by a route maintaining process, a route recovery process is executed. It reestablishes a path between a given source and the destination station. This process and the route maintaining process together have to overcome the problem associated with possible quick network topology changes in ad–hoc networks during a communication session.

Traffic Control Scheme: This scheme includes switching map management and routing processes at each node. The implementation should be kept simple and take into account low computing power and limited memory of each node.

Channel Reservation Scheme: A node sets up a connection between an incoming link and an outgoing link. This connection is virtual and is part of a virtual circuit. In such a connection, the node behaves as a virtual switch . After this connection is set up, the two links are marked as active, and cannot be reused for other virtual circuits. Because of the nature of wireless links, one link can be accessed by multiple nodes; thus the channel reservation scheme has to inform all nodes, which are able to access the same link, regarding the link usage.

Control Traffic Flood Minimisation Scheme: The route discovery process uses a flooding routing strategy to propagate a request within a network: a route request is rebroadcast until it reaches its destination. However, it is not always possible for a request to arrive at the destination successfully. For example, the destination may suddenly depart from the network. This can result in unnecessary control traffic flooding in the network. A minimisation scheme is used then to limit each control traffic flood. A packet of any kind should be eliminated if it has been travelling in the network longer than the predefined *timeout* without arriving at its destination. This procedure helps to minimise traffic control overheads caused by the flooding routing strategy.

Applying virtual circuit switching in ad–hoc networking is not a well-understood area. We decided to keep AVSR as simple as possible. Therefore, we can have a good control during its conceptual development. Functions and components of other existing protocols were carefully considered and adapted, allowing for understanding and ensuring the validity of components used in AVSR.

4.3 AVSR Design and Specification

AVSR has two major components: a MAC protocol for controlling how a node accesses the medium and a reactive routing algorithm for instructing a node how to obtain route information. Virtual circuit switching performs as an intermediate service between these two components. Furthermore, the control overheads of virtual circuit switching is integrated in the routing algorithm. The ideal situation for AVSR exists when nodes do not disappear during the whole communication session and there is no interference in the wireless medium. In this situation, the fundamental idea behind AVSR is that a given source node uses the reactive routing algorithm for obtaining the route to the destination node. After the route is found, the destination node sends the route information to the source. At the same time, intermediate nodes along this route use the MAC protocol to reserve virtual segments for the virtual circuit between the destination and the source node. Thus, data communication between the source and destination nodes are transmitted via a pre–planned route. Furthermore, a reserved and dedicated virtual circuit is set up for this communication session. Details of the MAC protocol and the routing algorithm are described in the following subsections.

4.3.1 MAC Structure

The success of virtual circuit switching is closely dependent on the medium access control. A MAC protocol that can provide virtual channels and allows channel reservation

is essential in AVSR. A virtual circuit between a given source node and destination node is established by a series of virtual channels. Reservations of these virtual channels ensures that no other medium access, from nodes that are not involved in the current communication session,is allowed. Thus, finding a MAC protocol that supports these requirements was our first task while designing AVSR.

The MAC protocol named Wireless–Channel-oriented Ad–hoc Multihop Broadband (W–CHAMB) [15] [30] was selected as the MAC component in AVSR. W–CHAMB is chosen because it is a MAC protocol that uses the "virtual channels" concept and is proposed for ad–hoc networks specifically. W–CHAMB has a built–in channel reservation scheme; therefore, no additional channel reservation scheme was needed.

W–CHAMB Structure W–CHAMB is a decentralised MAC protocol in which the packet transmission is channel–oriented. A channel between any two immediate nodes must be established before two nodes start transmitting data traffic. W–CHAMB functions in a decentralised manner because it was proposed for ad–hoc networks specifically. Medium access in W–CHAMB is based on existence of periodic frames. Each frame contains three types of time slots: *access channel* (ACH) slots, *traffic channel* (TCH) slots and *reservation minislots* (RMS). ACH slots are devoted to traffic control, and all stations have equal right to access them. TCH slots are used to carry data traffic only, and a given TCH slot can only be accessed by the end nodes that reserved it. RMS slots are used for channel reservation control and can incorporate error control. Figure 4.2 shows the structure of W–CHAMB frames.

In the original W–CHAMB proposal [15], there is only one ACH slot. We found it insufficient for supporting multiple data traffic streams, which have overlapping setup phases. For example, if two nodes want to obtain a channel at the same time, they can only use that ACH slot, so it results in a collision. To minimise this possibility, multiple ACH slots should be provided and nodes should use a random access scheme

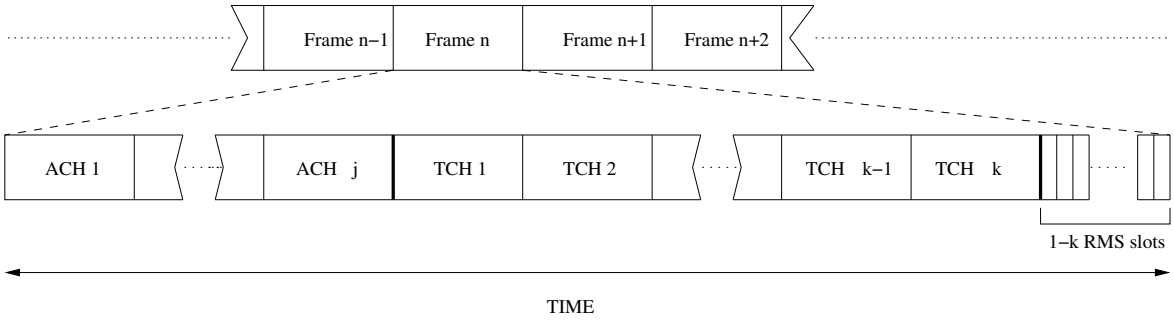


Figure 4.2: The W–CHAMB protocol frame structure. There are i ACH slots, k TCH slots and kRMS slots in one W-CHAMB frame. Note:j has to be less than k, and the number of TCH clost and the number of RMS slots have to be the same.

to access ACH slots. The number of TCH slots depends on how many parallel data traffic streams that network needs to support. More TCH slots are necessary in a busier network. RMS minislots are used for TCH slot reservations, so the number of reservation minislots has to be the same as the number of TCH slots. Additionally, all nodes in a network must have the same slot settings.

Duplex Channel Setup One TCH slot can be accessed by two nodes that are at either end of a link. A packet collision can happen when the two nodes access the TCH slot at the same time. In order to achieve a duplex communication without collisions, a TCH slot should be split into two smaller time slots. However, the two nodes still have to negotiate which sub–slot each of them can use. In our design, we assigned the first half TCH slot to the forward traffic stream[1], and the second half to the backward traffic stream[2].

Channel Duration The duration of both the ACH and TCH slots is same. In a wireless environment, the probability of having an error during the transmission is

[1]A communication stream flows from the source node to the destination node

[2]A communication stream flows from the destination node to the source node

proportional to the size of a packet[31]. Thus, the upper boundary of the size of a slot is limited by this factor. On the other hand, the minimum size of a slot has to be large enough to carry protocol headers and a sufficient amount of payload data[3]. There is no clear boundary for this factor; however, the proportion of the header length and the payload data in a TCH slot should be at least 1:1. A reservation minislot is used to carry one information indicating the corresponding TCH slot that is in use. Therefore, the size of such a minislot needs to be large enough for such information. The above issues have to be considered deeply before any practical implementation.

Channel Setup and Reservation Channel setup and reservation between two immediate nodes in W–CHAMB is a handshaking process. To start the process, the node that wants to establish a connection to another sends a request to that node via an ACH slot. The request should include the information regarding any TCH slots that are idle from the sender's point of view. The selection of idle TCH slots is done by sensing busy signals in RMS slots from the previous frame. After a node receives such a request, it will select one of the proposed TCH slots that is also idle from its point of view. The node then transmit a reply to the sender via the selected TCH slot. After the previous procedures, both nodes involved in this connection have to mark their corresponding RMS slots. These RMS slots should always be marked during the communication session. This operation ensures that any neighbour around a given communication zone will not try to use the selected TCH slot during this communication session. A successful example of setting up a channel between two immediate neighbours is shown in Figure 4.3. In this example, node A wants to set up a channel to node B. Node A first transmits a request to node B via an ACH slot. This request contains three proposed TCH slots, which are TCH 1, TCH 3 and TCH 5. These TCH slots have been sensed idle by node A because it did not hear any signals in

[3]Useful data that is carried by a packet.

their corresponding RMS slots sent from its surrounding neighbours. After node B receives the request, it needs to send a reply for completing the handshake. Because node B has heard a busy signal in RMS slot 3 from its neighbour node C previously, it removes TCH 3 from the proposal. Then node B randomly selects TCH 1 or TCH 5, and sends a reply back to node A via the chosen TCH slot, which is TCH 5 in this example. After this, the handshake between node A and node B is completed, both nodes have to keep generating a pulse at their reservation minislot 5 until the end of this communication session.

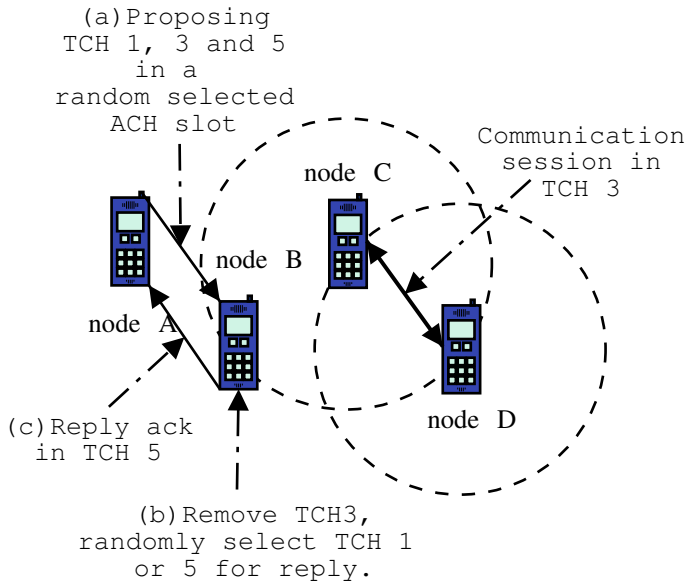


Figure 4.3: Channel setup and reservation in W–CHAMB.

Failure Handling A W–CHAMB's handshake will be unsuccessful if one of the following event occurs.

- *if* one of the nodes departs from the network suddenly; or
- *if* the node requesting a connection can not find TCH slots that are sensed idle to its point of view; or

- *if* the responding node can not agree on any proposed TCH slots; or
- *if* there is an error during packet transmission.

No active error detecting scheme has been implemented in our current design. Thus, a node can not detect a failure immediately after the failure occurs. The node requesting a connection determines an unsuccessful handshake if it does not receive acknowledgement within a predefined *timeout* period. It may start a new handshake if the need for that connection is still desired. The node responding to the request detects the failure if it does not receive traffic from the requesting node. After detecting the failure, the responding node stops using the TCH slot selected for this connection. However, it will not perform any recovery.

4.3.2 Routing Algorithm

A routing algorithm is used by nodes for obtaining route information. We chose to apply reactive routing for the routing component in AVSR. Furthermore, the procedures for setting up a virtual circuit were integrated into the *route discovery process*. As discussed in Chapter 3, this design can reduce control overheads generated by virtual circuit switching and the routing algorithm. Instead of building a routing algorithm from scratch, we adapted some functions from Dynamic Source Routing (DSR) to ease our design process. The corporation of switching map maintained by each node and the basic process related with routing are discussed in the following paragraphs.

Route Discovery Process: This process contains two main procedures: one for propagating a route request packet from the source node to the destination node and the other for sending a route reply packet from the destination to the source. A node, referred to as the source, wishing to discover a route to the destination generates a route request packet. It then broadcasts that request packet to its immediate neighbours in

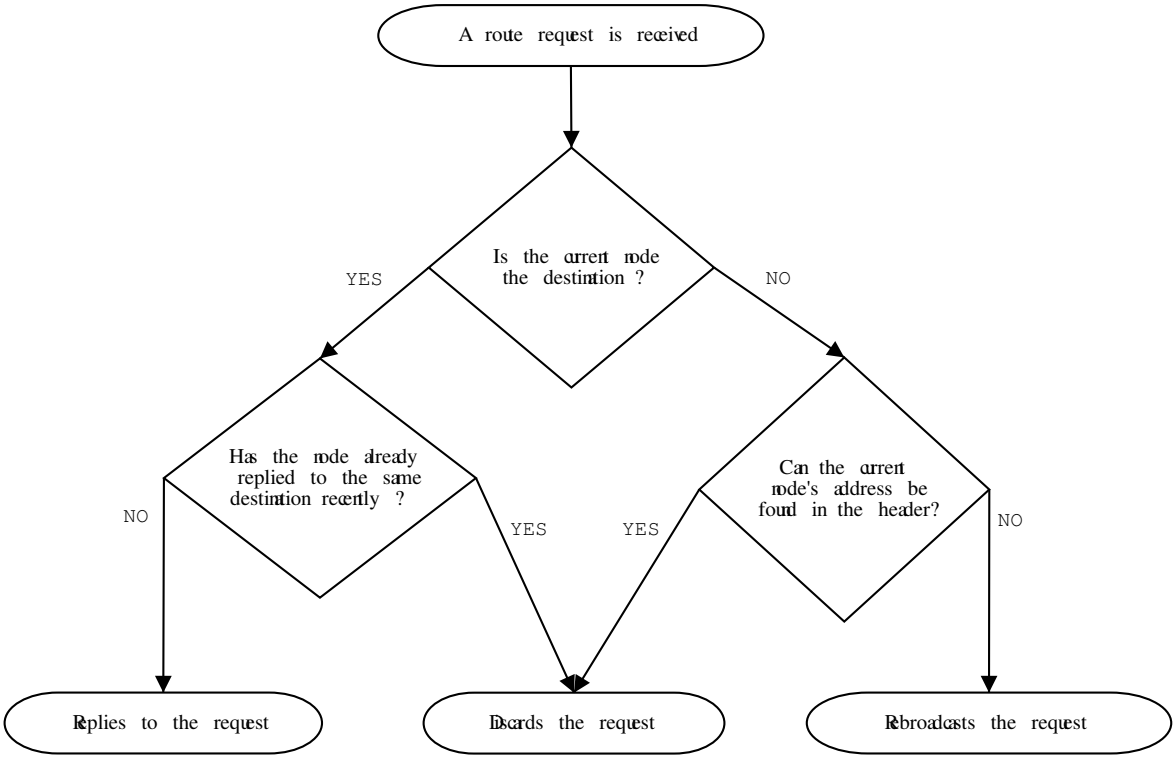


Figure 4.4: Decision making process at a node that receives a route request.

one ACH slot. When a node broadcasts a packet to all its immediate neighbours, it sets the receiving address of the packet to ALL. A node that receives a request packet can generate one of three possible responses: This decision making process of a node receiving a request is described graphically in Figure 4.4. and listed below.

1. If the node receiving the request is the destination, it checks the previous time when it received a request from the same source. A node only replies to a received request when the time difference between the previous and the current request packets, which were transmitted from the same source, is larger than the *reply timeout*. Otherwise, the received request will be discarded. This stops a node from replying to request packets initiated earlier by the same source for the communication session.

2. If the receiving node is not the destination for the request and its address is not included in the header of the request packet, it should append its address to the header and rebroadcast the receiving packet in an ACH slot.

3. If the receiving node is not the destination for the request and the address of the receiving node can be found in the header of the request packet, the receiving node has to discard the request.

When sending a route reply packet back to the source, DSR allows the destination to send the reply back to the source either via the reversal route that is included in the request header, or by initiating a new route request to the source with the reply attached to the new request. AVSR only uses the first method. Using the reversal route from the request results in obtaining a single bi–directional route between the source and destination. Thus, it makes no difference whether the source or the destination starts virtual circuit setup and also enables us to integrate the virtual circuit setup in the route reply process.

A route reply packet is transmitted in one randomly selected ACH slot. Unlike route request packets, a destination node or an intermediate node does not forward a route reply packet to all its immediate neighbours. A route reply packet being forward is received by a node, which becomes the next stop along the route to the source node. After the source node receives the route reply, it will start transmitting packets to the destination.

Each node, excluding the source node, involved in the route reply process is also responsible for setting up a virtual connection to the next node along the replying route. Before a node forwards a route reply packet, it initialises the channel setup handshake which was described in Section 4.3.1 and appends the channel setup request onto the route reply packet. After a node receives such a route reply packet, it detaches the request for virtual channel setup. The node examines the channel setup request and

tries to allocate a virtual channel based on the channel proposals in the request. If the virtual channel can be successfully obtained, the node then sends an acknowledgement for completing the channel setup handshake via the appropriate TCH slot and continues the route reply process. If the current node can not allocate a virtual channel for completing the handshake, the route reply packet will be discarded and this *route discovery process* will become unsuccessful. However, no error message will be reported to either the source node or the destination in our design.

A source considers its *route discovery process* as unsuccessful, if no acknowledgement is received from the destination within a predefined *route request timeout* period. Then the source will start a new *route discovery process* for obtaining routing information about the destination. During this new route request propagation towards the destination, the request may stop at intermediate nodes which have routing information and reserved channels, which are related to this communication session. If this situation occurs, the request informs the intermediate node to erase the routing information and release the channel. The number of times that a source node is allowed to restart a *route discovery process* has to be limited[4]. This limit stops a source node from obtaining routing information for a destination node that is not feasible to reach, and from generating unnecessary control traffic overheads. Such a case can occur:

- *if* the destination node departs the network; or
- *if* the network is overloaded and can not carry additional traffic; or
- *if* there are no intermediates between the source node and the destination node.

In our design, the destination node and intermediate nodes are not involved in detecting and recovering an unsuccessful *route discovery process*. However, each node (either an intermediate node or the destination node) is responsible for monitoring the

[4]See chapter 5 for further discussion

segment of a virtual circuit established by itself. It is possible that a given destination node and intermediate nodes may set up segments of a virtual circuit for an unsuccessful route discovery. Such a case occurs when the *route request reply* does not arrive at the source node but has partially travelled over its path successfully. This incomplete virtual circuit will not be used by any data traffic, because the source node can not send data traffic without receiving the *route request reply*. This results in the release of any incomplete virtual circuit after the timeout period. Additionally, nodes involved in this incomplete virtual circuit may also release their partial segment when they receive a new *route request* from the source. These two methods ensure no incomplete or idle virtual circuits are occupying resources in a network.

Route Maintenance Process: We use a simple process to ensure the validity of a route. The route from the source node to the destination node is maintained by the backward traffic stream from the destination node[5]. When backward traffic stream travels to the source node, each node along the path has to update the timestamp of the route to the destination node in its switching table.

A timestamp of a route is the time when the last successful transmission occurred by using the current route. Each route entry in a switching map must have a timestamp record. The "freshness" of a route can be determined by computing the difference between the current time and the timestamp of the route. The maintenance for the route from the destination node to the source node is done by a similar method. Instead of using backward traffic stream for updating timestamps of the acting route, it uses forward traffic stream from the source to the destination node.

Each node in the network maintains a *route maintenance process*. This process checks the validity (freshness) of each route entry in the switching map in the current node periodically. A route entry becomes invalid if the difference between its times-

[5]A route also means a virtual circuit here

tamp and the current time is greater than the *route timeout*. A invalid route entry in the switching map will be removed by the *route maintenance process*. The *route maintenance process* also informs the MAC protocol to terminate the current communication session and release the reserved channel. The changes of a route's validity is shown in Figure 4.5. A new entered route becomes expired after T time units, if its timestamp is not updated over this period. However, its valid period will be reset to T time units when its timestamp is updated.

The validity of a route entry is also checked by a node before the node uses the route entry for sending a packet. If the route entry for the destination node can not be found or has already expired, the current node has to abort the transmission and discard the packet. If a source node is forced to abort a transmission, it detects a route breakage, and will perform a route recovery process.

In our design, we assumed that a communication session has bi–directional data traffic. Therefore, intermediate nodes can receive data traffic from the source node and the destination node, and route entries for both directions can be updated and monitored. In the situation that the destination node does not send data traffic to the source node, the destination node has to generate acknowledgement traffic for maintaining the validity of the forward stream path.

Route Recovery Process: When a source node detects a route failure, it will issue a new route discovery process if the route is still required for a communication session. As described in the previous paragraph, a route failure can be detected by a node if:

- *if* the *route maintenance process* detects the timeout of a route;or
- *if* the node can not find a route entry for the destination node of a given packet.

Switching Map Management: In AVSR, each node keeps a switching map for traffic management. Each entry in the map consists of five components: source address,

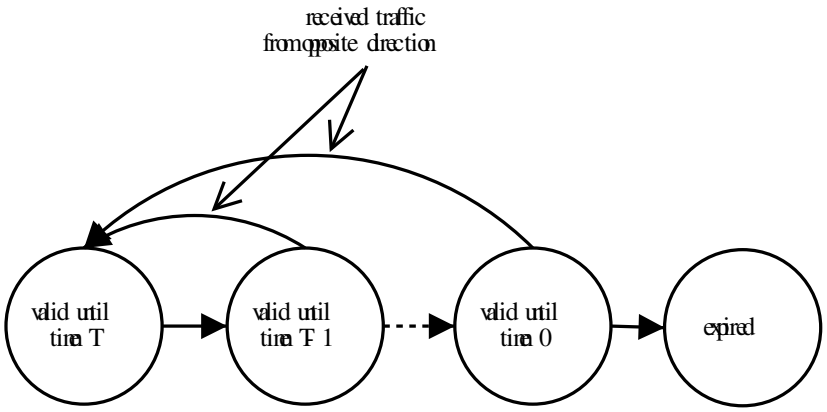


Figure 4.5: Changes of a route's validity at a node, where T means the time units before a route becomes expired.

destination address, traffic direction, timestamp and output TCH slot. The union of source address, destination address and traffic direction forms an unique identifier of each entry. This output TCH slot provides the information on how to forward a given packet. The timestamp is used to maintain the validity of the current entry as shown in Figure 4.5.

A new entry is inserted when a route reply packet is received. When a node receives a route reply, it selects an appropriate TCH slot from the proposal list and insert a new entry into its switching map. The traffic direction of the new entry has to be marked as *forward stream*. This node then sends the confirmation of its selected TCH slot to where the route reply was sent from. As a node receives a TCH slot confirmation, it inserts an entry into its switching map, and the traffic direction should be set as *backward stream*. In our proposal, traffic from the source to the destination is classified as *forward stream*, and the traffic in opposite direction is classified as *backward stream*.

Virtual Circuit Reservation and Release: As soon as a node inserts a new route entry into its switching map, the node has to start the reservation process for the TCH slot in the new route entry. The method of reserving a channel is described

on page 48. A node should reserve all TCH slots that are to be incorporated in its routes. It prevents other nodes in its neighbourhood from accessing these TCH slots. The reservation of a TCH slot shall be cancelled after the *route maintenance process* detects that its corresponding route as being broken.

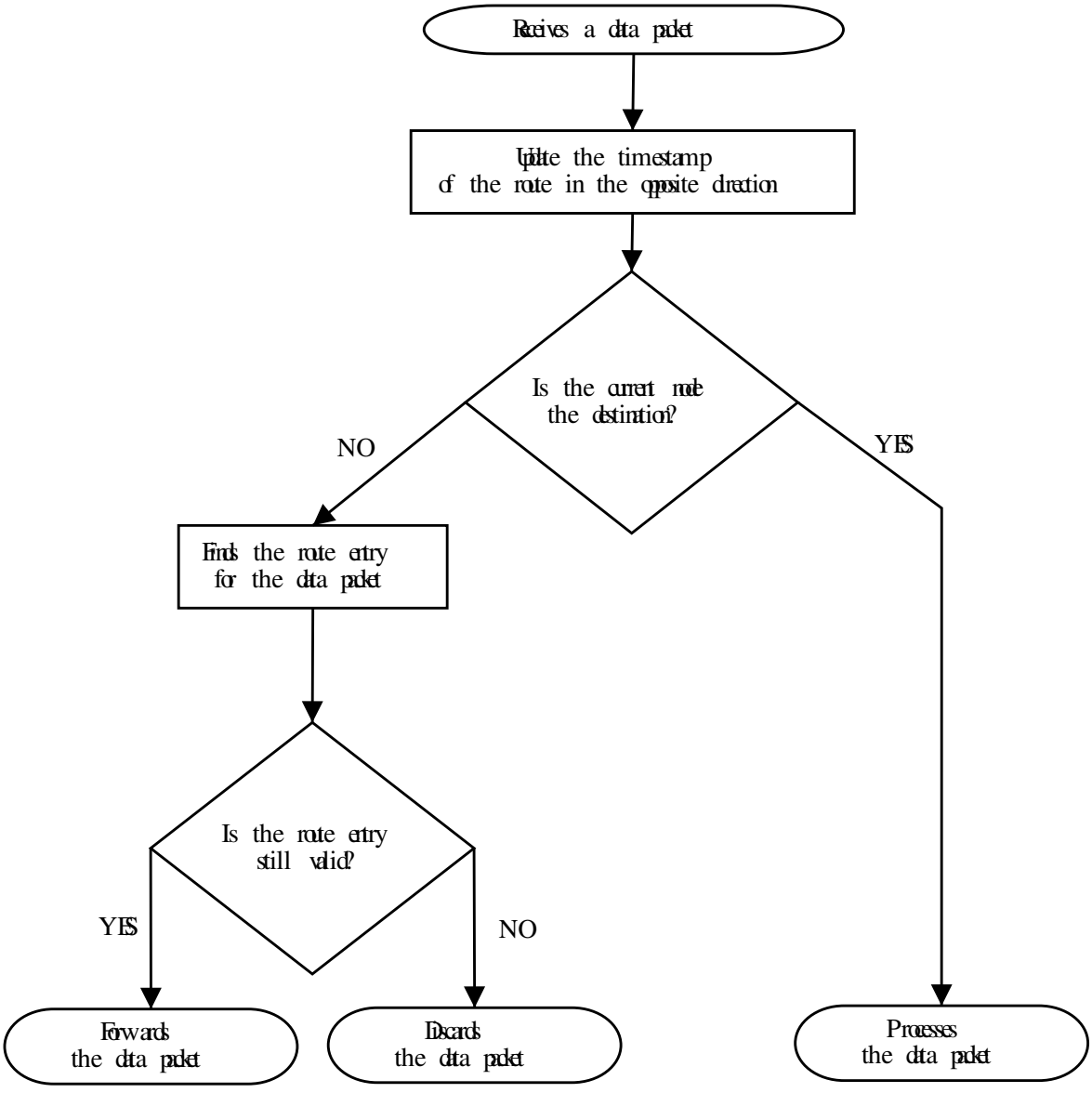


Figure 4.6: Decision making process by a node which receives a packet.

Incoming Data Packet Processing: The flowchart explaining how a node processes a received data packet is shown in Figure 4.6. As a data packet arrives at a node, the node firstly updates the timestamp of the route that is used for the traffic stream in the opposite direction. The node then examines if it is the destination of the packet. If the current node is not the destination for the received packet, the node has to find the route entry in its switching map for routing the packet onward. The found route has to be examined for its validity before a data packet is scheduled for retransmission in the next TCH slot that is associated with the current route entry in the switching map. If the found route is not up–to–date, the packet should be discarded. On the other hand, if the current node is the destination node for the packet, it will process the packet based on its content.

Optimisation Issues: The purpose of developing AVSR is to demonstrate virtual circuit switching with reactive routing in an ad–hoc network. Our proposal was focused on its feasibility rather than its performance and efficiency. In DSR, there are optional features to improve the performance. For example, DSR enables a node to cache routing information from overheard route requests and allows intermediate nodes to reply to route requests when it has a caching route to the requesting destination node. This method increases the performance and efficiency of route discovery in DSR. However it is not adopted in the AVSR discussed in this thesis because it would make virtual circuit setup more complex. Such features have been left for further development and are discussed in Chapter 7.1.

4.4 AVSR Demonstration

A simple ad–hoc network, as shown in Figure 4.7, is provided as an example network for our AVSR demonstration. There are six nodes in this network, which are connected

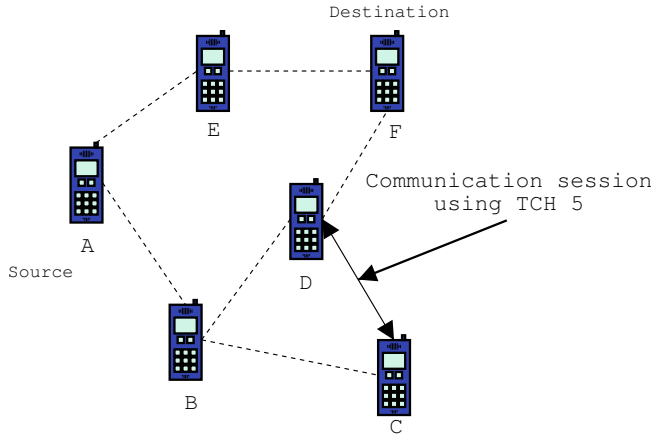


Figure 4.7: A simple ad–hoc network for AVSR demonstration.

by wireless links. Any two nodes that are reachable by each other are linked by dotted lines. Bidirectional links are assumed. In this network, there is one existing communication session between node C and D. TCH slot 5 is occupied by this session.

ILet us assume that node A wants to set up a new communication session with node F. Node A starts a route discovery process by broadcasting a *route request* packet to its immediate neighbours via a randomly selected ACH slot. This request will be propagated to node F by intermediate nodes. Intermediate nodes involved in this propagation will put their addresses into the header of the request packet in order to complete the route information. During the process of the *route request* propagation, node D can receive multiple requests from both node B and node C. However, requests received after the first one will not be rebroadcast because of the limitation of control traffic flooding. Similarly, node F can receive requests from both node D and node E, but only the request which arrived first will be replied to. In this example, a request travelling via the path of A→E→F should arrive at node F before other request packets travelling via other paths because this path the shortest path.

After the first request, which is forwarded by node B, arrives at the destination, node F initiates a *route reply* packet which includes the route information obtained

from the header of the received request. It also appends the list of TCH slots that are idle from its point of view to the request. In this example, TCH 5 would be excluded because node D, an immediate neighbour of node F, is using TCH 5 for its communication session with node C.

This *route reply* will be sent back to node A via the path of F→E→A, which is the reversal path travelled by the first received request. Instead of been broadcast to all immediate neighbours, reply packets are transmitted to a specified node. To start the reply process, node F sends the reply to node E via a randomly selected ACH slot. When node E receives the reply, it firstly needs to confirm the TCH slot for the communication between itself and node F. This is done by randomly choosing a TCH slot from the proposed list and sending an acknowledgement via that TCH slot. Having confirmed the TCH slot, node E inserts an entry for forward traffic stream from node A to node F into its switching map and the output TCH slot of the inserted entry is set to be the same as the previous confirmed TCH slot. Node E also initiates reservation of that TCH slot after entering the route entry into its switching table. Meanwhile, node F, having received the acknowledgement from node E via a TCH slot, inserts a route entry for backward traffic stream from node F to node A into its switching map and the output TCH slot is set the same as the incoming TCH slot of the acknowledgement. Node F then begins the channel reservation process for that TCH slot.

Before node E forwards the reply to node A, it has to change the proposed list of TCH slots in the reply. This is done by swapping the proposal list with another list which contains idle TCH slots for node E. The reply packet is then forwarded to node A via a randomly selected ACH slot.

Node A, having received the reply from node E, uses the same procedure as described above for confirming the TCH slot for communication between node A and node E. This confirmation will also be used by node E for setting up the route entry in

its switching map for backward traffic stream from node F to node A. When node A inserts its route entry to node E into its switching map and starts the channel reservation procedure, the whole *route discovery process* is completed. As the *route discovery process* is completed, all nodes should know complete routing information and have reserved TCH slots for a given communication session. Thus, node A can start the communication session by transmitting the traffic in the selected TCH slot.

During the communication session, node E is responsible for relaying traffic from node A to node E. Thus, if node E departs from the network, it will cause a link failure in the virtual path used for communication between node A and node F. The *route maintenance process* at node A should detect this failure after it does not receive any traffic from node E for a time interval of one *route timeout*. After a link failure is detected, node A starts a new *route discovery process* for resumption of the current communication session. During the session recovery, node A buffers all traffic addressed to node F. The communication session will be resumed after node A completes the new *route discovery process* successfully. In this example, node A should be able to obtain another route to node F, say via node B and node D.

The current design of AVSR has a passive *channel reservation release scheme*. Therefore, the virtual path for the communication session between node A and node F will be kept reserved, even when node A terminates its communication session. When such an event happens, all nodes involved will not receive any traffic and will not be able to update their route entries related to this communication session. The *route maintenance process* in theses nodes will detect link failures of the routes related to the session. No *route recovery process* will be started because it was node A that terminated its communication session to Node F. All reserved channels for this session should be released after a link failure is detected.

4.5 Summary

This chapter introduced the basic features of AVSR. We discussed the design and specifications of AVSR. AVSR has two components: a MAC protocol and a reactive routing algorithm. Virtual circuit switching is used for linking them together. W–CHAMB was chosen as the MAC protocol, and we adapted some functions of DSR for our routing algorithm. A example was given for showing how a node uses AVSR for obtaining the route information, setting up the virtual circuit and recovering from a link failure. In the next chapter, we will discuss how we constructed a simulation model for testing and evaluating AVSR.

Chapter 5

Simulation Modelling for AVSR

In the previous chapter, we discussed the design issue and specifications of the AVSR. To evaluate the performance of the AVSR, we built a network simulator, which simulates behaviour of the AVSR and an existing protocol, DSR, with which AVSR is compared. The simulator was implemented in the*Akaroa 2* package, which uses the technique of Multiple Replication In Parallel (MRIP) for speeding up the simulation process. This technique speeds up simulation by launching independent replications on multiple computers. Therefore, more observations can be collected during a given time interval than running a single replication on one computer within the same period of time.

The network simulator was implemented by using the paradigm of Object–Oriented Design (OOD). Such a design gives the simulator good flexibility and extensibility. These are important factors for our simulator because the performance of the analysed protocol, AVSR, should be studied taking into the account of its various features.

This chapter is structured as follows. The framework of the simulator is firstly presented. Then we discuss of the issue of credibility of simulation results and the method used for speeding up simulation processes. Use of *Akaroa 2* for enhancing the credibility of our simulation results is discussed at the end of this chapter.

5.1 Simulator Framework

The simulator was developed for studying the mobility and communication activities in a small ad–hoc network, which contains a number of wireless nodes within a restricted area. We used object–oriented design, which gave us good flexibility during the developing process. It also allowed us to model entities in an ad–hoc network as individual objects.

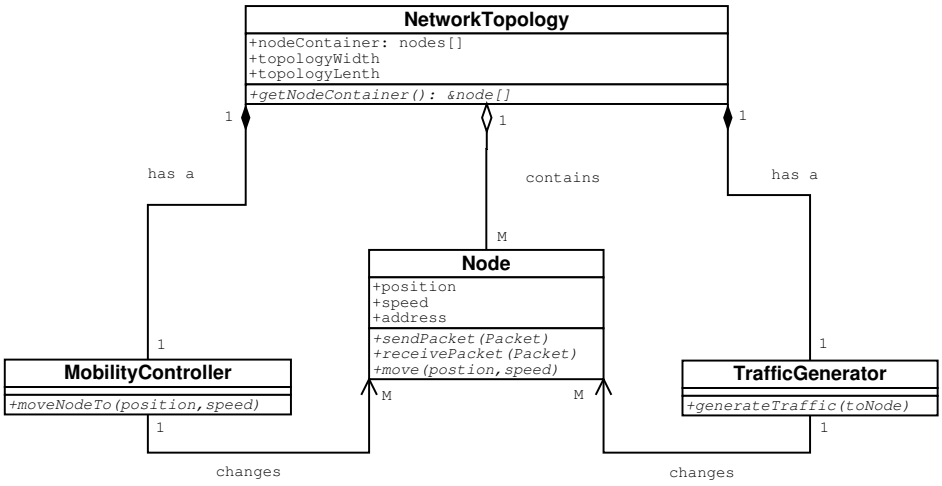


Figure 5.1: An UML class diagram represents the framework of our simulator.

There are four main classes in our simulator, and they are named **Node:**, **NetworkTopology**, **MobilityController** and **TrafficGenerator**. An UML class diagram of our simulator is shown in Figure 5.1 and its components are discussed in the following.

Node Class One primary class in our simulator is the **Node**. Thus, we built a simple model for it, as shown in Figure 5.2, for studying its activities in an ad–hoc network.

A node in our simulator can play in two different roles: begin either a **sender** or a **receiver**. It is also involved in **mobility management** activities when it is participating in either role. When a node operates as a sender, it has to understand

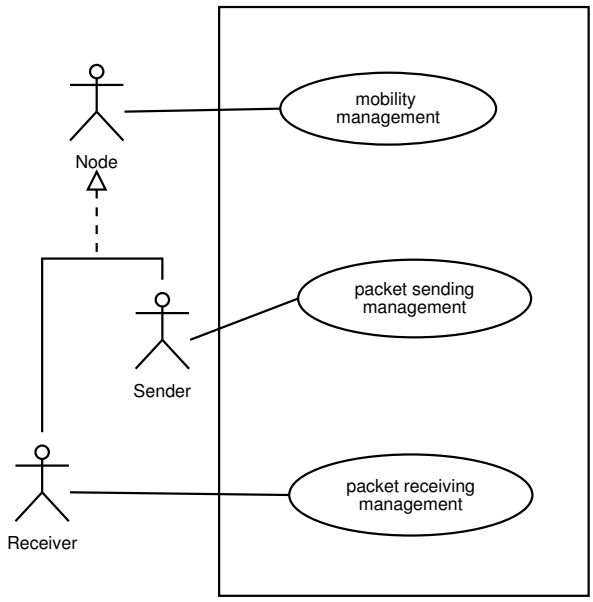


Figure 5.2: An UML use case diagram represents activities of a **Node** instance in an ad–hoc network.

instructions for **packet sending management**. Similarly, a node has do understand **packet receiving management** when it behaves as a receiver. These management functions can be formulated as follows:

- **Mobility management process** instructs a node how to change its location in a network. Primary parameters for mobility management are the coordinates of the destination location and the travelling speed of the node.

- **Packet sending management process** governs the process of sending a packet from one node to another. A node can use different methods for sending different types of packets. The functions belonging to this management process are formulated by protocols used in the network, such as MAC and routing protocols.

- **Packet receiving management process** specifies the operations at the node after it receives a packet. A node can react differently when it receives different

types of packets. This management process is also formulated by MAC and routing protocols used in the network.

NetworkTopology Class: The **NetworkTopology** is a container for storing **Node** instances in a network. It also defines the geographical properties of the network topology, such as the size and shape.

There is only one **NetworkTopology** allowed in one simulation model but it can have multiple **Node** instances in it. The **NetworkTopology** also has one **MobilityController** and one **TrafficGenerator**. These two objects change the properties of **Node** instances when simulation is in progress.

MobilityController Class: The **MobilityController** is responsible for the movement processes in the network. The **MobilityController** obtains the geographical properties of the network topology from its owner, a **NetworkTopology** object. Such information ensures that the **MobilityController** does not allow a **Node** instance to move to a location that is outside the geographical boundary of the network.

TrafficGenerator Class: The **NetworkGenerator** is responsible for the communication processes within a network. The **NetworkTopology** instance uses a **TrafficGenerator** for generating communication streams in a network. A communication stream is a series of packets between two selected nodes during a defined period of time.

The**MobilityController** and **TrafficGenerator** implement the movement and communication models of the simulation. Thus, these classes have to be overwritten for modelling a specific simulation model. A discussion of the movement and communication models of our experiments are presented in Chapter 6

5.2 Credibility of the Simulation Output

There are two issues that have to be addressed for ensuring the credibility of a simulation experiment [20]. These issues are discussed in the following paragraphs.

Application of appropriate elementary source(s) of randomness: Simulation studies of telecommunication networks can require long runs for obtaining their final results with an acceptable statistical error. Therefore, it is critical that a simulation uses a pseudo random number generator (PRNG) that does not repeat its sequence during the simulation run. Furthermore, recent studies of pseudo random numbers suggests that the number of pseudo random numbers from one PRNG used in a single simulation should be restricted. For example, if one conducts a simulation experiment which involves two dimensional uniformity and uses a PRNG with cycle length L, then one should not use more than $8\sqrt[3]{L}$ numbers from this PRNG during that simulation [5].

Appropriate analysis of simulation output data: Any stochastic simulation study has to be treated as a statistical experiment. Therefore the statistical error must be reported with the experiment results, otherwise they represent realisations of random numbers. The statistical error provides the degree of confidence in the accuracy of a given result. The most common method of error assessment is measuring the confidence interval of a result at a given confidence level. A creditable simulation should have a narrow confidence interval at a high confidence level. It is known that the width of the confidence interval will shrink as the number of collected output result increases [20]. However, it can be a time consuming task for collecting a large number of output results from a complex simulation model, taking a long time to generate sufficiently many output results in the first place.

In order to ensure the credibility of our simulation results and to shorten simulation

time, we decided to use the *Akaroa 2* package.

5.3 *Akaroa 2*

Akarao 2 package [6] has being developed and evaluated by the Simulation Research Group at the Department of Computer Science, University of Canterbury. It is a simulation controller designed for sequential control of random processes during a simulation. The PRNG used by the *Akaroa 2* is a Combined Multiple Recursive PRNG, which has a cycle–length of approximately 2^{191}. This PRNG had been evaluated statistically and proved to be a good source of pseudo random numbers [4],

As discussed previously, collecting observations for a creditable simulation can be time consuming. *Akaroa 2* uses the technique of *multiple replications in parallel* (MRIP), as shown in Figure 5.3, for speeding up the simulation process. In this approach, one *Akaroa 2* controller maintains multiple *engines*. During a simulation run, the *Akaroa 2* controller launches independent simulation model on each engine attached to it and assigns each of them a different sequence of random numbers. Using non–overlapping sequence of random numbers for parallel simulation replications ensures that no correlation between outputs from different processes exists. Output date from different replications are reported to a global analyser at the controller. The controller analysis collected outputs regularly until the error of the output reaches the satisfying level. Then all simulation engines are instructed to stop their simulations.

One difficulty with conducting a simulation experiment is to determine the runtime of a simulation, which can give an output with an acceptable statistical error. However, it is not possible to find an exact runtime for reaching an acceptable statistical error before a simulation starts. Traditionally, one runs a simulation for a fixed time and then performs the data analysis. If the statistical error of the output is greater than the desired level, then one has to repeat the whole process again and hopes the error

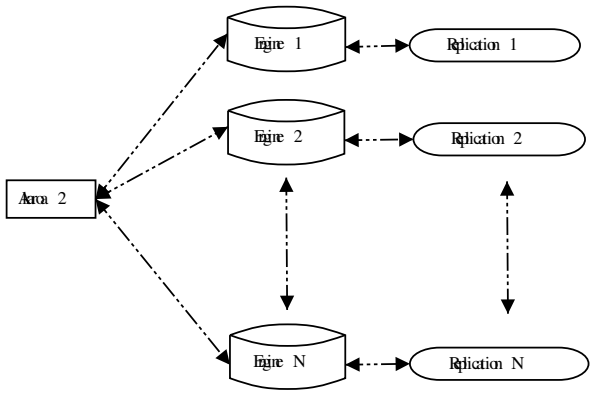


Figure 5.3: The *multiple replications in parallel* technique with N simulation engines as implemented in the *Akaroa 2*

to reaches a satisfactory level in the next time. Such an approach can be exhaustive when repetitions are performed manually. A simulation governed by *Akaroa 2* does not suffer such a problem, because the data analysis is performed by the controller automatically at consecutive checkpoints, until error drops to its expected value.

5.4 Summary

In this chapter, we briefly discussed the architecture of our network simulator for evaluating the performance of the AVSR. This simulator is designed in a Object–Oriented manner, which gives us a great flexibility for further expansion. Furthermore, this simulator was designed to interact with the simulation controller *Akaroa 2*. With the assistance of *Akaroa 2*, we are able to ensure the credibility of our simulation results. In the next chapter, we will discuss our experiment methodology and experiment results.

Chapter 6

Performance Evaluation of AVSR

We used stochastic simulations for evaluating the performance of AVSR in an ad hoc network. The results then were compared with those obtained from the ad hoc network operating with DSR over CSMA as its traffic control strategy.

In this chapter, the configuration of our simulation models is firstly described. The experimental methodology and the numerical results obtained are then presented, and followed by the chapter summary.

6.1 Simulation Model Assumptions and Configurations

The overall goal for our experiments was to assess the dynamic behaviour of AVSR in an ad–hoc network. We also wanted to compare the performance of AVSR with another protocol which uses the same routing strategy as AVSR but employs datagram switching as its switching technique. DSR over CSMA was selected as a reference model, because DSR is where AVSR adapts its routing strategy from and CSMA is a MAC protocol supporting datagram switching only (see discussion in Chapter 3). Such a comparison should indicate the difference between virtual circuit switching and

datagram switching in an ad hoc network.

Two simulation models were constructed by using the simulator discussed in Chapter 5. These two models have been developed under identical assumptions, network topology, mobility model, traffic generation model, and wireless environment assumptions. In the rest of this chapter, these two models will be referred as **AVSR model** and **DSR–CSMA model**.

6.1.1 Common Configurations and Assumptions

In this subsection, we will discuss the assumed network configurations used in experiments. These include network topology, mobility model, traffic model and wireless environment assumptions.

Network Topology: In both simulation models, wireless terminals are able to move randomly and freely in a 40 meters by 30 meters rectangle closure, as shown in Figure 6.1. Such an area is similar to one floor in the Mathematics and Computer Science (MaCS) building at University of Canterbury in Christchurch. We chose such an area because it is representative for our working environment and help us to visualise the network topology during the development.

Mobility Model: The movement pattern of each node in our simulation models was governed by the 'random waypoint' model [10]. Initially, each node is placed at a random position in the network. As the simulation progresses, each node randomly selects a destination and chooses a velocity between one to two meters per second. It then moves to the destination at the selected speed. In our model, nodes do not have any pause time before they start moving to their new positions. Therefore nodes are continuously moving in the network.

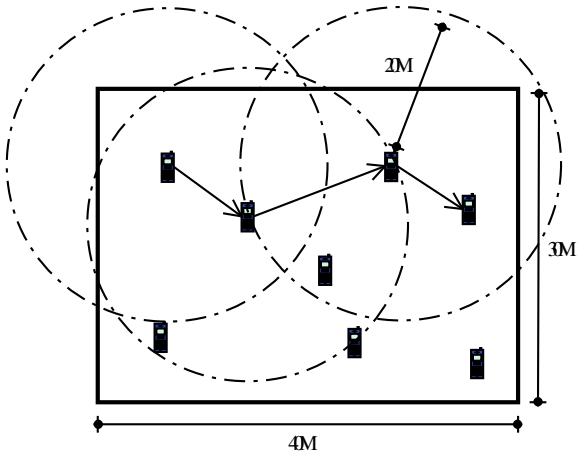


Figure 6.1: Network Topology model for simulated networks.

Traffic Model: We assumed peer–to–peer communication for modelling traffic flows in our simulation model. Each node waits for a time interval before it starts a new communication session. These time interval are driven by a Poisson process and the mean inter–event time is set to 60 seconds. When a node is ready to start a new communication session, it randomly selects a destination node within the network, and then begins a new communication session. Each communication session lasts exactly for one minute. During this period, a node uses another Poisson process to generate traffic packets and the packets generation rate is set to 300 packets per second. The destination node has to send a reply packet when it receives a traffic packet from the source node. These reply packets represent the backward traffic flow from the destination node. All packets transmitted during a communication session have the same size, which is determined by the half duration of a TCH slot.

We set the duration of a TCH slot to 45 μsecs in our AVSR model. After removing all guarding gaps, the duration of a TCH slot is 40 μsecs. In a wireless medium which provides transmitting speed of 11Mbps, such a TCH slot can carry 400 bits per slot. When packets generation rate of a traffic generator (referred to as a source node) is

300 packets per second, the node will generate traffic at a rate of 60,000 bit per second. We assumed that a destination node replies every incoming packet sent by a traffic generator. Thus, the amount of data travelling from a destination node and source node shall be 60,000 bit per second approximately. We assumed such a traffic scenario for representing a duplex voice conversation which requires 128Kbps bandwidth.

Wireless Environment Assumptions: The following assumptions specify the wireless environment used in our simulation:

- The speed of the wireless medium is 11Mbps.
- The transmission range of a node is set to twenty meters. Thus, multi–hop communication is forced to occur in our model.
- There are perfect wireless channels. There is no data loss during packet transmission.
- The propagation delay is so small that it ca be negligible.
- Each node allows its neighbours to use it as an intermediate node.
- Nodes do not experience power shortage during the simulated runtime of their performance.
- Each node has an unique network address and knows addresses of all other nodes within the network.
- All nodes are synchronised to the same time and are capable to use W-CHAMB.

6.1.2 AVSR Model Configurations

AVSR is traffic control protocol which covers both routing strategy and medium access control. The configuration of both components are discussed in the following

paragraphs.

Medium Access Control Setting: As discussed in Section 4.3.1, one W–CHAMB frame consists of three types of time slots, and they are called ACH, TCH and RMS slots, respectively. The number of time slots of each type in one W–CHAMB frame is adjustable for different applications. We configured our model to have eight ACH slots, sixteen TCH slots and sixteen RMS slots in one frame. Such a configuration theoretically is capable to support eight concurrent communication initialisations and sixteen concurrent communication sessions when reuse of time slots is not considered.

The duration of a ACH slot, a TCH slot and a RMS slot was set to be the same as described in [31], and they are 45 μsecs, 45 μsecs and one μsec, respectively. Within an ACH slot or TCH slot, two μsecs at each the beginning and the end are used as guarding gaps. Furthermore, the 23rd μsec of a TCH slot is also used as a guarding gap in order to achieve duplex channel access (see Section 4.3.1).

The *reserved channel timeout* period used for determining channel reservation and link failures was set to 10 $m$secs. This value is also used as the *route timeout* parameter. If a node does not receive any traffic which uses a particular route within the following *route timeout* period, it determines that route is broken. Thus, the node should release the channel which is reserved for the broken route.

Routing Strategy Settings: One major parameter in the routing strategy of AVSR is *route request timeout*. This parameter is used for determining whether a *route discovery process* is successful or not. In our AVSR model, *route request timeout* is set to 500 $m$secs, which was also assumed in [1] and [10].

In the design of AVSR (see Section 4.3.2), the *route timeout* parameter is used when controlling the validity of a route in the *route maintenance process*. We set this parameter is set to 10 $m$secs. Such a value was selected because of the setting of the

traffic model. In our traffic model, the mean inter–event time is 3.3 *m*secs. We decided to set the *route timeout* three times larger than the mean packet inter–arrival time. Thus, a *route maintenance process* will have low probability of confusing false link failure with long time interval between two packets.

AVSR uses the *route request retry* parameter to control the number of *route requests* which are allowed to sent by a source during its *route discovery process*. In our AVSR model, this parameter is set to 3 times. Based on this setting, it will takes 1.5 seconds for a source node to send three *route requests* without receiving an acknowledge (three *route request timeouts*). If another *route request* is allowed, the waiting time will take longer than two seconds, which exceeds the industry standard for maximum call setup delay[7].

Another important parameter in AVSR is *route request packet lifetime*. We set this parameter to the duration of three hops in our network. Based on our topology (see Figure 6.1), three hops communication is sufficient enough to connect any two given nodes within the topology.

Table 6.1 summaries all parameters and their values used in our AVSR model.

6.1.3 DSR–CSMA Model Configurations

Our DSR–CSMA model uses CSMA for its medium access control and DSR as its routing protocol. We used the standard IEEE802.11b specification to configure our CSMA protocol. The only modification we made was setting the maximum packet size to 200 bits, which is the maximum amount of data which can be carried by one TCH slot in a duplex communication. Such a modification should give this model a similar traffic pattern as in the case of AVSR. Furthermore, the collision avoidance mechanism (RTS/CTS) is not implemented here, because it would cause too much overhead for sending packets which are short.

Parameter	Value
W–CHAMB frame setting	ACH: 8, TCH: 16, RMS: 16
ACH slot duration	45 μsecs
TCH slot duration	45 μsecs
RMS slot duration	1 μsecs
channel reservation timeout	10 $m$secs
route request timeout	500 $m$secs
route timeout	10 $m$secs
route request retry	3 times
route request packet lifetime	3 hops

Table 6.1: Parameter configurations of AVSR.

The configurations of DSR–CSMA were set identical to the routing configurations of AVSR. Thus it allows us to perform a fair comparison of AVSR and DSR–CSMA.

6.2 Experiment One: All nodes generate traffic

Experiment One was the first experiment conducted during our evaluation, and its methodology and the results are presented in the following subsections. The results of this experiment show weaknesses of the AVSR protocol, but are not able to indicate the cause of it. Thus, we designed a refined experiment, Experiment Two, to determines the cause. Discussion related to Experiment Two can be found in Section 6.3.

6.2.1 Experiment Methodology

This experiment was designed to assess the performance of AVSR and DSR–CSMA models. We started the experiment by setting the node population in both models to

five nodes. After measurements form both model were collected, we kept increasing the node population by one node until it exceeded twenty nodes. As discussed in Section 5.3, we used *Akaroa 2* package to ensure the credibility of collected results.

Node Population The range of node population was selected on the basis of informal simulation runs[1]. Based on such informal runs, we found that both AVSR and DSR–CSMA feature poor connectivities when the node population is less than five nodes. We were not interested in the performance of a network with a low connectivity, therefore we set the initial node population to five nodes.

Our informal simulation runs also show that,the performance of AVSR became extremely poor after the node population exceeds twenty nodes. Therefore we decided to stop this experiment when the node population reached twenty nodes and expected the results would give us an explanation.

Overall Traffic Load In this experiment, all nodes are set as traffic generators that they are able to start new communication sessions. Thus, increasing the node population not only strengthens the node connectivity, but also raises the traffic load in the network. Such a design introduces two variables to be assessed in this experiment. Experiment of such degree of freedom has created an additional difficulty for determining causes of performance degradation of AVSR. Therefore, we performed another experiment, Experiment Two, described in Section 6.3.

Simulation Runtime We conducted informal test runs for determining the real time that should be simulated. These tests runs were set for 5, 10, and 15 minutes real time, and there results show no significant difference in performance. Thus we decided to limit our simulations to 5 minutes of real simulated time.

[1]By an informal simulation, we mean a single replication during which precision of results was not analysed.

6.2.2 Performance Measurements

The performance of either the AVSR or DSR–CSMA model were measured by the metrics listed below. These estimates of metrics were collected at the end of each simulation run.

- *Packet Delivery Success Ratio* (PDSR) – the ratio between the number of packets generated by sources nodes and the number of acknowledgements received by source nodes. This metric indicates the ability of the network successfully transmitting a packet in a communication session after its route or virtual circuit is established.

- *Route Discovery Success Probability* (RDSP) – the probability of a source node completing a *route discovery process* successfully. It is measured by the ratio of the number of successful route discoveries and the number of route discoveries issued. A new communication session can only be started after its virtual circuit or route is established successfully. Therefore this metric indicates the difficulty of setting up a new communication session in a network. During this experiment, a RDSP is classified as low if it is less than 50%.

- *Data Packets Collision Probability* (DPCP) – the probability of collision occurring when transmitting a data packet. This measurement shows the stability of a traffic stream in a virtual circuit.

- *Control Packets Collision Probability* (CPCP) – the probability of collision occurring during the transmission of a control packet. This metric shows the difficulty for setting up a route or virtual circuit. Note, that when CPCP assesses a large value then the corresponding RDSP will be small.

In each simulation, multiple replications were performed in order to collect estimates, which are consistent with our required precision error and confidence level.

Metric	***Precision Error*** (%)	***Confidence Level*** (%)
Packet Delivery Success Ratio	0.05	95
Route Discovery Success Probability	0.05	95
Data Packets Collision Probability	0.10	95
Control Packets Collision Probability	0.10	95

Table 6.2: Desired precision and confidence level for each metric.

The maximum acceptable precision errors and confidence levels of the metrics are listed in Table 6.2. We set DPCP and CPCP to have high precision error because it took an extremely long time to obtain a simulation result with even so high precision error. However, we claim that 10% precision error is still acceptable in our experiments.

Nodes	PDSR		RDSP		DPCP		CPCP	
	AVSR	DSR	AVSR	DSR	AVSR	DSR	AVSR	DSR
5	0.974953	0.898983	0.774625	0.842270	0.011263	0.006389	0.092848	0.021249
6	0.969391	0.901674	0.790514	0.883706	0.020711	0.008965	0.148141	0.032556
7	0.962979	0.898556	0.796152	0.895908	0.031431	0.011471	0.202618	0.043547
8	0.961569	0.892665	0.771563	0.906179	0.035720	0.014488	0.259461	0.054943
9	0.959790	0.888132	0.755002	0.909497	0.040611	0.017522	0.307970	0.067566
10	0.959347	0.884983	0.732620	0.911337	0.041349	0.019732	0.353320	0.076712
11	0.955838	0.888348	0.699039	0.916393	0.048994	0.021827	0.395916	0.090836
12	0.956879	0.876458	0.670251	0.911520	0.049090	0.025163	0.435685	0.104330
13	0.957295	0.877630	0.645732	0.910946	0.049544	0.026420	0.470531	0.113500
14	0.956376	0.869337	0.606590	0.908878	0.051967	0.029725	0.505233	0.128350
15	0.958775	0.858459	0.579990	0.902139	0.047186	0.033123	0.535605	0.141877
16	0.956937	0.859735	0.547781	0.900562	0.051039	0.035129	0.564335	0.155883
17	0.957344	0.855575	0.527073	0.894064	0.051912	0.036747	0.589648	0.166892
18	0.955636	0.853411	0.507801	0.890026	0.053560	0.039966	0.612899	0.183526
19	0.957822	0.841376	0.476087	0.890423	0.050506	0.043513	0.635137	0.196156
20	0.959618	0.844546	0.449424	0.878961	0.047012	0.043269	0.655675	0.214198

Table 6.3: Numerical results of Experiment One. Note: PDSR = Packet Delivery Success Ratio; RDSP = Route Discovery Success Probability; DPDP = Data Packet Collision Probability; CPCP = Control Packet Collision Probability.

6.2.3 Numerical Results

The numerical results of Experiment One are shown in Table 6.3. These results characterise the performance of AVSR and DSR–CSMA. They are further discussed in Section 6.2.4.

Packet Delivery Success Ratio (PDSR) Based on our experiment results, both AVSR and DSR–CSMA have high packet delivery success ratio. The minimum PDSR for AVSR and DSR–CSMA are 0.95 and 0.84, respectively. However AVSR constantly has higher PDSR than DSR–CSMA. A plot indicating the trend of PDR in AVSR and DSR–CSMA as the number of nodes increas is shown in Figure 6.2.

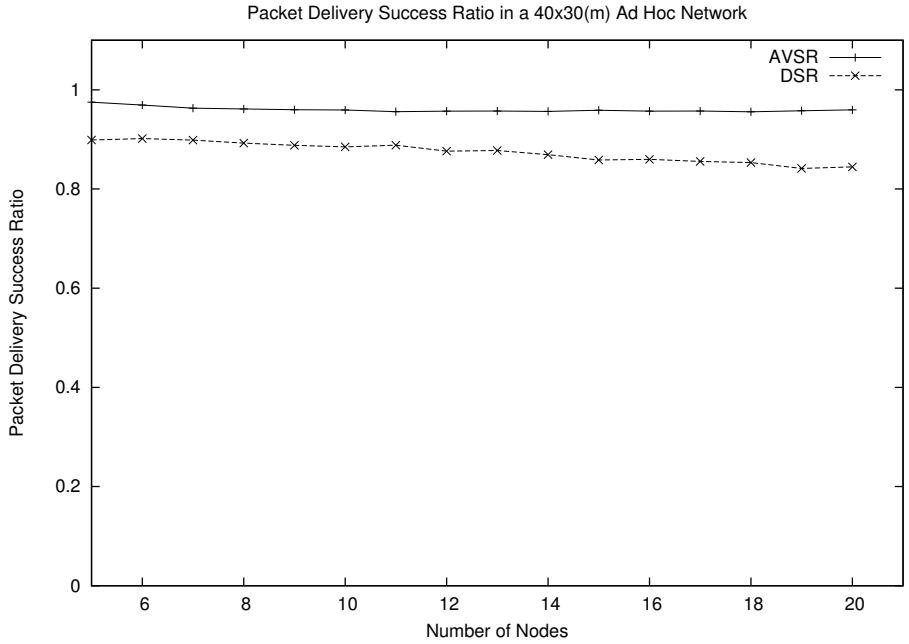


Figure 6.2: Packet Delivery Success Ratio under AVSR and DSR–CSMA.

Route Discovery Success Probability (RDSP) Figure 6.3 shows the RDSP in AVSR and DSR–CSMA. The initial value of RDSP under AVSR for five nodes is 0.775. It has a positive correlation until it reaching its maximum of 0.796 for 7 nodes. After

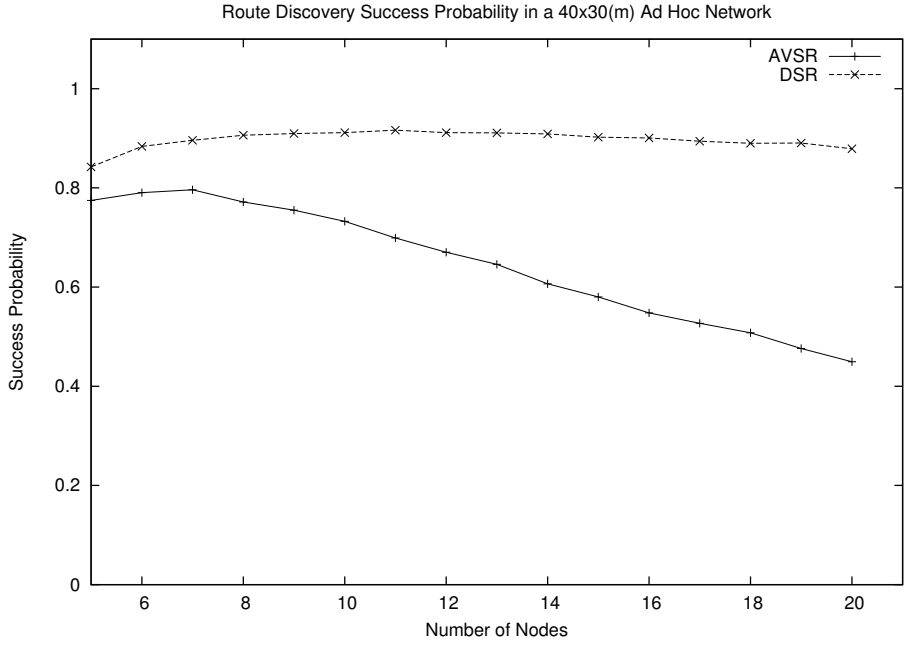


Figure 6.3: Route Discovery Success Rate under AVSR and DSR–CSMA.

that maximum, RDSP decreases. When the node population reaches 20 nodes, the RDSP in AVSR drops to 0.499, which is approximately 60 percent of its maximum. The AVSR line in Figure 6.3 clearly shows that RDSP in AVSR has a decreasing trend as the node population increases.

In contrast, RDSP under DSR–CSMA is much more stable than under AVSR. The difference between its maximum and minimum value within the investigation range of nodes is 0.07. RDSP under DSR–CSMA model starts at 0.852 for 5 nodes and increases until it reaching 0.916 for 11 nodes. This is the maximum of RDSP under DSR–CSMA. After this maximum point , RDSR under DSR–CSMA decreases, but the decreasing rate is much smaller than the rate under AVSR. The DSR curve in Figure 6.3 shows that the RDSP in DSR–CSMA case is higher than in the AVSR case. It also indicates that RDSP under DSR–CSMA is more stable than under AVSR as the node population increases.

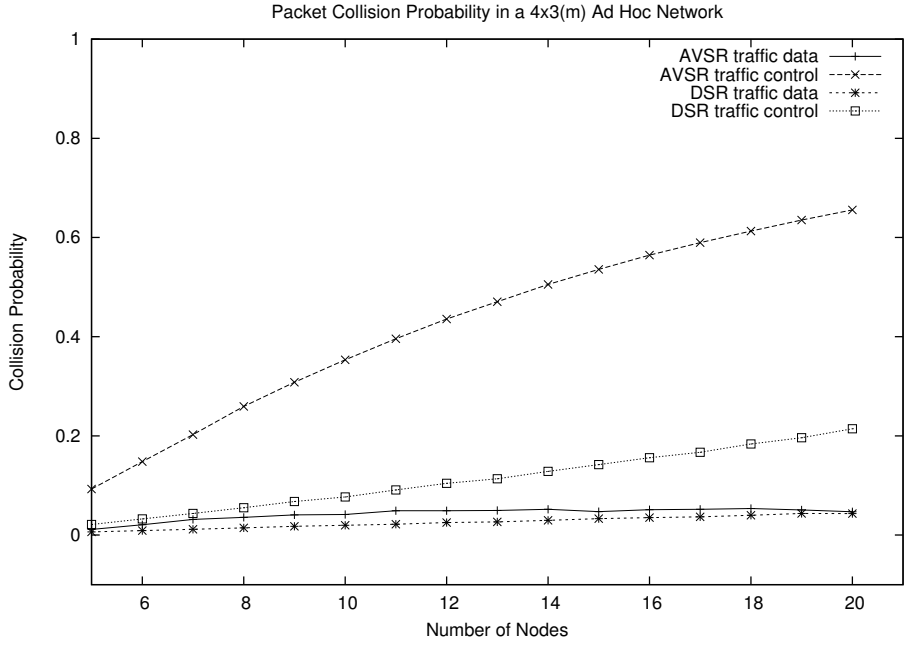


Figure 6.4: Packet Collision Rate of AVSR and DSR.

Data and Control Packet Collision Probability (DPCP and CPCP) Figure 6.4 shows the DPCP and CPCP under AVSR and DSR–CSMA. This figures shows that packets used for traffic control have higher collision probability than packets used for carrying data. It also shows that both DPCP and CPCP are larger under AVSR than under DSR–CSMA.

In both AVSR and DSR–CSMA networks, the DPCP and CPCP increase as the node population raises and CPCP is constantly higher than DPCP. Such a result indicates that node population has a stronger influence on CPCP than on DPCP.

6.2.4 Discussion of Results

In both our simulated models, we assume that the networks have perfect wireless channels. As a result, the only errors that occur are packet collisions. This assumption caused that our experiments overestimates the performance quality. However, the re-

sults determined for AVSR and DSR–CSMA would have the same bias, which weakens their neutral comparison relevant.

Performance of Established Communication Sessions: Our experiment shows that both AVSR and DSR–CSMA have high Packet Delivery Success Ratio (PDSR) regardless of thier size of node population. This indicates that the performance of a running communication session is not affected by the node population in either case. However a network with a high PDSR cannot guarantee a good overall performance. Based on our definition of PDSR, as discussed in Page 79,it can be seen as the packet throughput of a established communication session. Therefore, we can only claim that both AVSR and DSR–CSMA can provide stable service if a communication session is successfully established.

Scalability: Data Packet Collision Probability (DPCP) in both models remain lower than 6% as the node population increases, but their Control Packet Collision Probability (CPCP) does not. When the traffic density increases, AVSR has a higher CPCP, which causes a lower Route Discovery Success Probability (RDSP). As a result, AVSR has scalability issues when the node population increases.

RDSP indicates the resistance for setting up a new communication session in the network. Therefore, it is difficult to access a network if it has a low RDSP. Our result shows that AVSR has a lower RDSP than DSR–CSMA. This is caused by high Control Packet Collision Probability (CPCP) under AVSR.

As evident from Figure 6.4, the rate of increase of Control Packet Collision Probability (CPCP) under AVSR is much more rapid than under DSR–CSMA. Consequently, when the node population raises, the loss of control packets under AVSR is higher than under DSR–CSMA. Under both DSR–CSMA and AVSR, the success of one *route discovery process* depends on a series of control packet deliveries. Any control packet loss

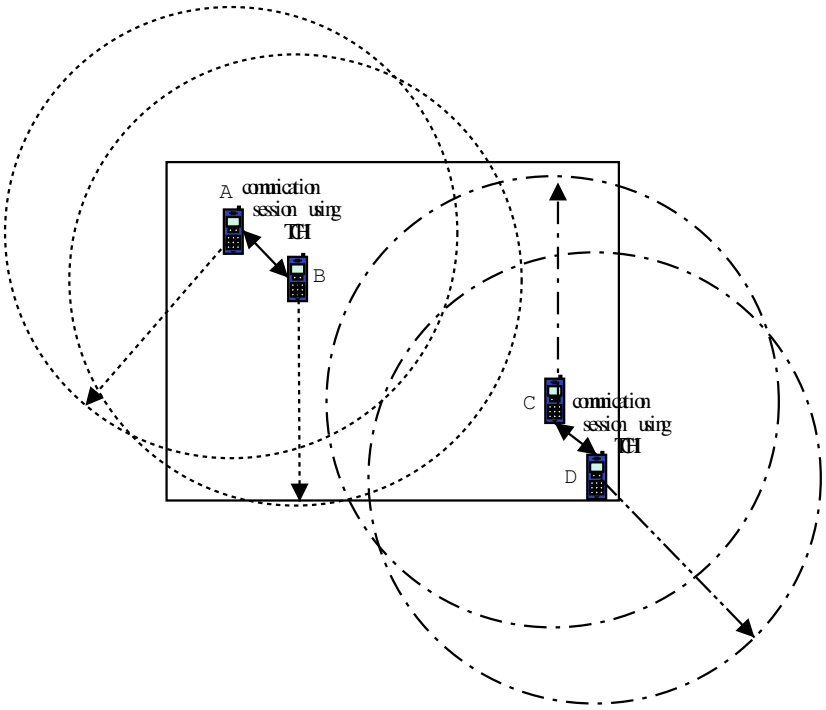
in the *route discovery process* causes it to fail. Therefore, a model with a high CPCP has a low RDSP.

CPCP in DSR increases as the node population raises, but its rate of increase is much less than under AVSR. Our result shows that the maximum CPCP in case of DSR–CSMA is only one third of the maximum CPCP in the case of AVSR. Under DSR–CSMA, both traffic control and traffic data packets shares whole bandwidth. In contrast, AVSR uses only one third[2] of whole bandwidth for transmitting traffic control packets. This limited bandwidth under AVSR leads to a bottleneck for supporting a large number of traffic control packets. As a result, the performance of transmitting traffic control packets, measured by CPCP, under AVSR model is inferior than under DSR–CSMA. In Experiment One, the number of traffic control packets increases as the node population rises. Therefore, we can not distinguish whether the increase of traffic control packets is caused by the increase of node population or the increase of overall traffic load (see Page 78).

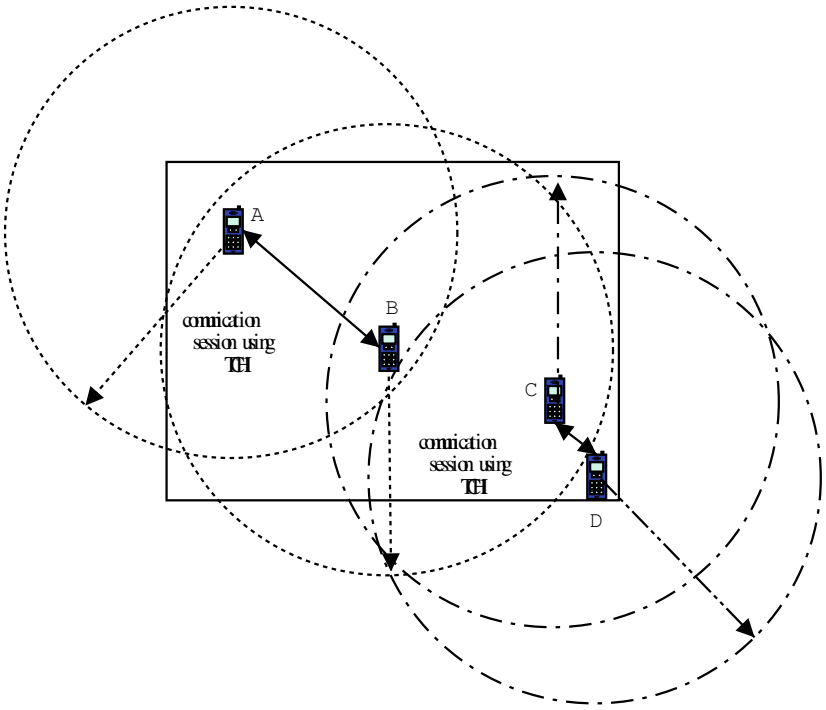
Intruding Node Problem: In AVSR, we used W–CHAMB as our medium access control scheme. W–CHAMB is a channel–oriented protocol, and is expected to reduce Data Packet Collision Probability(DPCP). However, our experiment results show that we do not have a lower DPCP under AVSR than under DSR–CSMA. After examined traces of simulations event under AVSR, we found out that packet collisions were caused by the intruding node problem. A intruding node under AVSR is a wireless terminal that moves into its neighbour's communication range and uses a TCH slot, which is also used by itsneighbour.

An example of intruding node problem is shown in Figure 6.5. It is assumed that nodes can move freely within the rectangle closure and the transmission range of each node is represented by a circle surround it. As shown in Figure 6.5(a), there is a com-

[2]the proportion of ACH slots in one W–CHAMB frame



(a) Two communications using the same TCH slot without interfering each other.



(b) Node B moves into the communication range of node C and becomes an intruding node.

Figure 6.5: An example of an intruding node problem.

munication session between node A and node B, and another communication session between node C and node D. Both communication sessions are using the same TCH slot. However there is no interference between two communication sessions because their transmission range are not overlapped with each other. In Figure 6.5(b), node B moves to a new location where is within the transmission range of node C. As a result, interference occurs. This causes high collision probability for both communications sessions.

The MAC protocol in AVSR, W–CHAMB, assigns an unused time slot to each communication session during its session setup. However, W–CHAMB can not identify the intruding node problem actively. When a intruding node problem occurs, communication sessions will constantly have degraded performance until the intruding node disappears. In the other hand, the intruding node problem also occurs under DSR–CSMA, but its impact is much smaller than under AVSR. This is because that the MAC assumed here, CSMA, actively prevents a node to transmit data when the medium is busy.

One major finding from Experiment One is that the accessibility of a network under AVSR reduces when its node population increases. A network has low accessibility when its RDSP is low. Experiment One shows that the RDSP under AVSR becomes lower when the node population increases. This experiment also indicates the correlation between CPCP and RDSP. Our analysis recommends that the design of the MAC layer under AVSR can not accommodate heavier control traffic as the node population increases. Therefore, it is difficult for a node to complet a *route discovery process* under AVSR when its node population becomes higher.

In Experiment One, we assumed that all node in the network are traffic generators. Such an assumption gives the network a heavier overall traffic load when it has a higher node population. Therefore, we can not determine whether the accessibility issue of

AVSR is caused by high node population or heavy overall traffic load. Experiment Two is designed to determine which of these two factors is responsible for the low accessibility issue under AVSR model, when the network has larger population of nodes.

6.3 Experiment Two: A subset of nodes generates traffic

Experiment Two is a refinement of Experiment One. This was designed to identify the factor which is responsible for the low accessibility issue under AVSR. Therefore, we only performed simulation runs of AVSR based networks.

6.3.1 Methodology

In this experiment, the number of traffic generators (NTG) was set first to one. We then measured the performances of the AVSR models when its node population was 5, 10, 15 and 20 nodes. The whole process was repeated four times, and NTG in the model was increased by one before a new iteration starts.

Node Population Similar to Experiment One, we measured the performances of our models when their node population were between 5 to 20. However, on each occasion the node population was increased by five. This change was made because we found that the change of a model's performance was not noticeable when its node population is increased by one or two nodes only. Thus, collecting data for every node population is not time efficient during a experiment.

Overall Traffic Load Overall traffic load in a network depends on the number of traffic generators (NTG) in the network. A network that has many NTGs also has high

overall traffic load. The number of traffic generators (NTGs) is a controlled factor in Experiment Two. We decided to set the range of NTG between one to five, and to vary the number of all nodes in the networy from 5 to 20.

Simulation Runtime Experiment Two used the same simulation setting as Experiment One. All simulation runs were set to simulate 5 minutes of real time.

6.3.2 Performance Measurements

In Experiment Two, we are interested in the cause of the poor accessibility under AVSR. Therefore, we only conducted simulations which showed poor accessibility in Experiment One, and measured Route Discovery Success Probability (RDSP) and Control Packet Collision Probability (CPCP).

When we first started Experiment Two, we set each simulation run to estimates with 95% confidence level and 5% precision error. However, such settings did not give us interpretable figures. The results were very random, so they would lead to unreliable conclusions; see Appendix A. To improve the Credibility of our results, the required confidence level and precision error for all results were set to 99% and 1%, respectively.

6.3.3 Numerical Results

Table 6.4 shows the numerical results of Experiment Two. These results are further discussed in subsection 6.3.4.

Route Discovery Success Probability The resulted RDSP as a functon of NTGs is plotted in Figure 6.6. There are four curves in this figure. Each line represents results collected for a particular node population.

As shown in Figure 6.6, AVSR always has a higher RDSP when its node population is high. Furthermore, all curves in this figure are practically flat. Such flat curves

Nodes	NTG	RDSP	CPCP
5	1	0.755677	0.087289
	2	0.749730	0.094549
	3	0.745750	0.090070
	4	0.732442	0.097531
	5	0.742490	0.088624
10	1	0.705260	0.355823
	2	0.718851	0.353374
	3	0.720335	0.355556
	4	0.705121	0.354743
	5	0.724382	0.355748
15	1	0.586743	0.532719
	2	0.575717	0.534515
	3	0.572548	0.532694
	4	0.589664	0.534942
	5	0.579534	0.534227
20	1	0.463755	0.654634
	2	0.443076	0.654576
	3	0.446497	0.654368
	4	0.437879	0.655516
	5	0.443542	0.655950

Table 6.4: Numerical results of Experiment Two. Note that NTG, RDSP and CPCP mean Number of Traffic Generators, Route Discovery Success Probability and Control Packet Collision Probability, repressively.

indicate that the number of traffic generators in AVSR is independent of the Route Discovery Probability (CPCP).

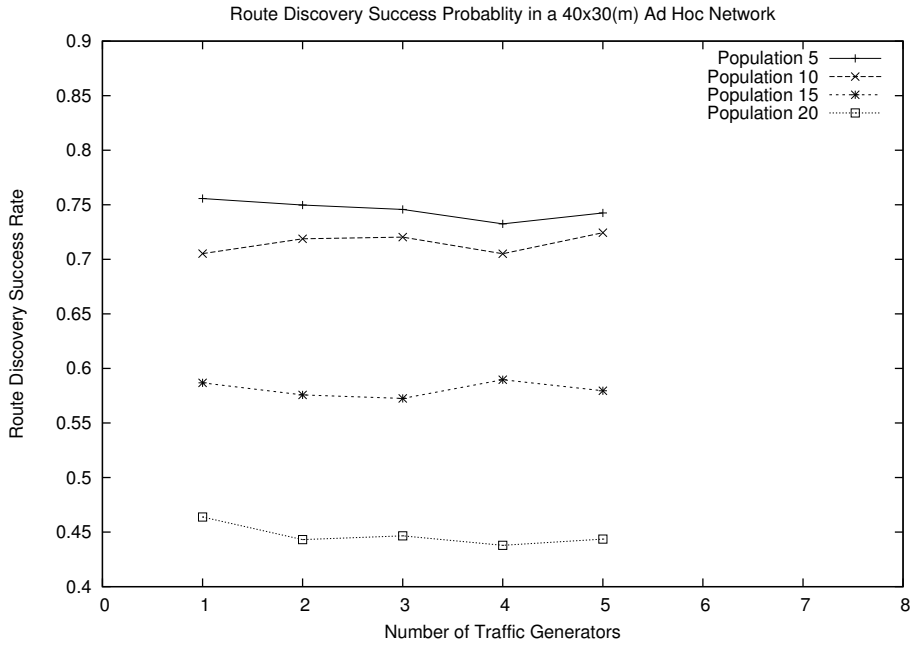


Figure 6.6: Route Discovery Success Probability under AVSR.

Control Packet Collision Probability The relationship between number of traffic generators, node population and CPCP under AVSR is shown in Figure 6.7. There are four curves in this figure. Each of them represents CPCP for a particular node population.

The figure also shows that CPCP under AVSR increases when the node population rises. Furthermore, all curves in the figure are practical flat. Such fact indicates that CPCP under AVSR does not increase when the NTG rises. Thus, NTG and CPCP are not correlated.

Control Packet Collision and Route Discovery Fail Probabilities Our results show that there is a correlation between CPCP and RDSP. To show it, we calculated

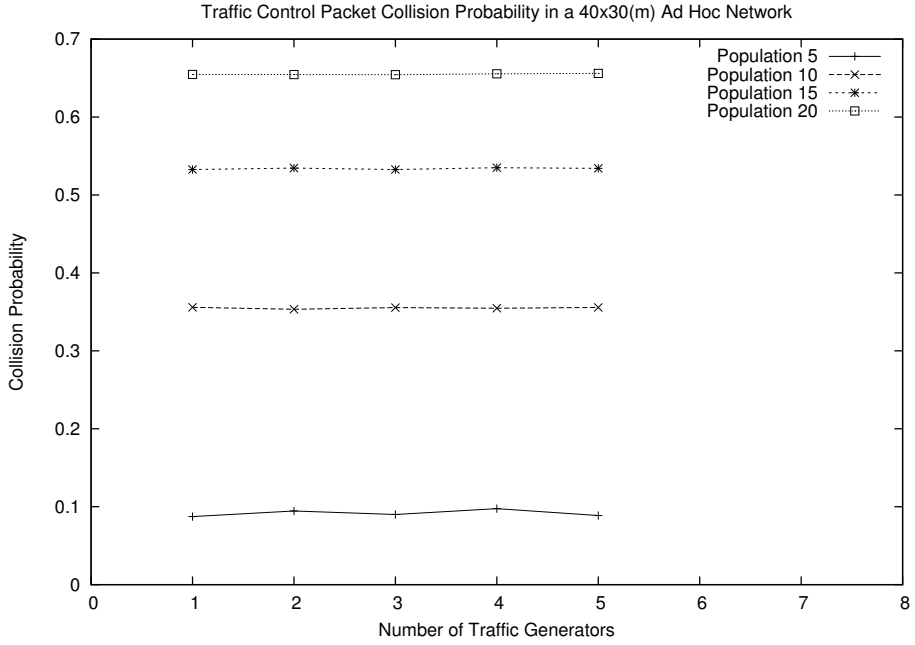


Figure 6.7: Control Packet Collision Probability under AVSR.

the reciprocal of RDSP, which has positive correlation with CPCP. The reciprocal of RDSP is Route Discovery Fail Probability, and is referred as RDFP in this experiment.

As discussed previously, either RDSP or CPCP is correlated with NTG under AVSR. Therefore, we only looked at the maximum value of CPCP and RDFP for each node population, for showing their relationship in the worst case.

Figure 6.8 shows a positive correlation between CPCP and RDFP. It also shows that CPCP has a stronger positive correlation with RDFP when the node population is higher than ten. Both RDFP and CPCP increase when the node population rises from p=5 to p=20.

6.3.4 Discussion of Results

Based on results of Experiment Two, we can confirm that the number of traffic generators (NTG) in an AVSR based network is not responsible to the low accessibility

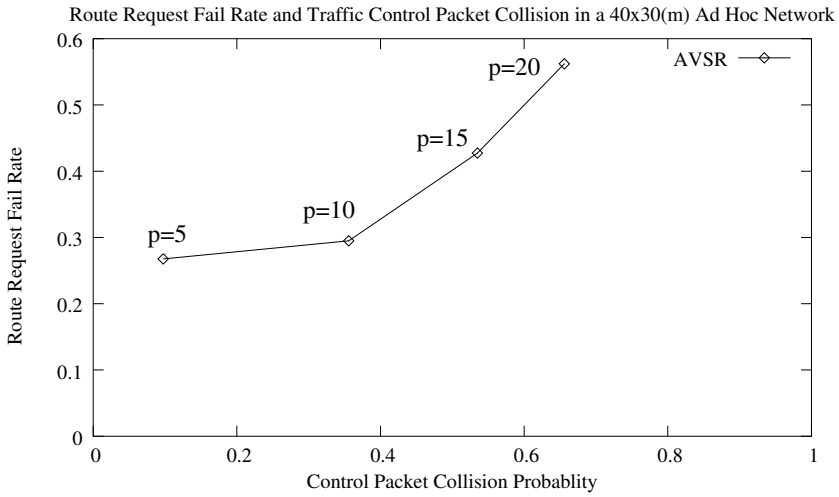


Figure 6.8: A Positive correlation of CPCP and RDFP under AVSR.

indicated in subsection 6.2.4. It is the increasing node population causing the performance degradation under AVSR.

Under AVSR protocol, *route request packets* are routed following the flood routing strategy. Thus, more flooding traffic is generated when a node has more immediate nodes. Under AVSR, a node is expected to have more immediate neighbours when the network has a larger node population. As a result, an AVSR based network has more flooding traffic when it has a larger node population. To support more flooding traffic, more bandwidth for traffic control packets is desired.

The AVSR protocol is an implementation dependent protocol. It is not possible to resize the bandwidth dynamically for traffic control packets in an AVSR based network after it is implemented. Thus, such a network has a low CPCP when its ACH slots are not sufficient for accommodating flooding traffic produced during a *route discovery process*. As discussed in subsection 6.2.4, the success of one *route discovery process* depends on a series of control packet deliveries. Any control packet loss in the *route*

discovery process causes it to fail. Therefore, this experiment shows AVSR protocol has low RDSP when it has high CPCP.

6.4 Summary

In this chapter, we evaluated the AVSR protocol. Our evaluation results indicates that it is feasible to use this protocol in ad hoc networks. However, its performance can be outperformed by a network using a datagram switching technique, which is DSR over CSMA.

Our evaluation shows that critical weaknesses of the AVSR protocol. One of them is its accessibility that reduces when the node population increases. Based on the current design of the AVSR protocol, the fixed bandwidth can not accommodate the increasing flooding traffic, which is produced by *route discovery processes*, when the node population in the network rises. To neutralise this weakness, the MAC layer of the AVSR protocol should be reviewed and redesigned. However, such research would out of the scope of this research. Recommendations for further developments are discussed in Chapter 7.1

Chapter 7

Conclusions and Future Works

In the thesis, we have proposed and evaluated a new traffic control protocol, named Ad–hoc Virtual Switching Routing (AVSR). It is a protocol that demonstrates the feasibility of implementing virtual circuit switching in ad–hoc networks.

So far, virtual circuit switching has not been consider as a switching technique for ad–hoc networks. Traditionally, an alternative technique, known as datagram switching, is selected for building ad–hoc networks. This switching technique is well known as being efficient for transmitting short messages, but not messages consisting of larger number of packets. In contrast, virtual circuit switching is considered as more efficient for supporting communication sessions which consist a large number of packets. Additionally, transmission delay in over a virtual circuit is more predictable than a datagram, and bandwidth for each communication session can be assigned dynamically during setting up a virtual circuit. These properties provide a friendly platform for implementing Quality of Service at the MAC layer. These advantages motivated us to investigate the feasibility of implementing virtual circuit switching in ad–hoc networks.

The main problem with implementing virtual circuit in ad hoc networks is that it requires application of special MAC protocols. Namely, such a MAC protocol has to

operate in a distributed manner and be accessed by time slots or frequency channels. However, there was no MAC protocol which fulfills these requirements until recent technology advancement. Thus, we are now able to conduct this feasibility study of implementing virtual circuit switching in ad hoc networks.

In our research, we selected Wireless–Channel-oriented Ad–hoc Multihop Broadband (W–CHAMB) as the MAC layer of AVSR. It was chosen not only because it is capable of supporting virtual circuit switching but also its built in channel reservation scheme. This saved us from investigating a new scheme for reserving established virtual paths.

The routing strategy of AVSR protocol is based on the Dynamic Source Routing (DSR) protocol because of the simplicity and easy implementation in DSR. AVSR uses a modified version of the *route discovery* process of DSR. Such a modification enables AVSR to set up a virtual path between the source and destination nodes during its *route discovery* process. The other functions of DSR are ported into AVSR without any modification.

Evaluation of AVSR was done by using stochastic simulations. *Akaroa 2* package was used for controlling each simulation run. It speeds up the simulation runtime by launching multiple replications on computers on a *local area network*. Additionally, it ensures the credibility of the results in a simulation. The evaluation results show that the accessibility of a network using AVSR reduces, when its node population increases. The bandwidth allocated for traffic control packets is fixed in AVSR. As a result, it can not accommodate the increasing flooding traffic, which is produced by *route discovery processes*, when the node population rises in the network. To neutralise this weakness, the MAC layer of AVSR should be reviewed and redesigned. However, such a research is out of the scope of this thesis.

Our research investigated different issues of applying the virtual circuit switching technique in ad–hoc networks. The results of our experiments indicate that the vir-

tual circuit switching technique is applicable in ad–hoc networks, although weaknesses have been identified in AVSR. Overcoming these weaknesses can be a goal of future investigation.

7.1 Future Works

The AVSR protocol, which we proposed, uses the flood routing strategy for its *route discovery process*. One issue with using such a routing strategy in ad–hoc networks is caused by generation of overheads associated with flooding of traffic during the routing process. In the flooding routing strategy, the amount of flooding traffic is proportional to the size of a network. As a result, a network using the flood routing strategy for its traffic control suffers a performance degradation when its node population increases. Such an issue is clearly identified during our protocol evaluation process and discussed in Chapter 6.

In order to reduce the overhead produced by the *route discovery processes* in AVSR, a mechanism which efficiently removes unnecessary flooding traffic is desirable. During our background study, we identified a few techniques that address this issue. These techniques include using geographical information by each node during the *route discovery processes* (see [12] and [17]), caching overheard routing information which is used as an optional feature in DSR (see [11]), and using a directional or smart antenna for radio transmissions (see [16]). These techniques should be considered in future developments of the AVSR protocol.

We used stochastic simulation for evaluating our protocol. In our simulation model , it is assumed that the simulated network operates in an ideal wireless environment. Therefore, multi–path fading, radio transmitting errors, capture effects and other wireless channel characteristics were not modelled. However, such an environment does not exist in real life. It is our intention to build a more realistic simulation model in future

studies.

Appendix A

Credibility of Simulation Results

In Experiment Two, we were not able to collect reliable results for the metric, Route Discovery Success Probability (RDSP) at low confidence level and high precision error. As presented in Figure A.1, the line for population 15 shows no trend and should be considered as random lines. To improve the Credibility of our results, we reduced the statistical error of our simulation by increasing its confidence level to 99%.

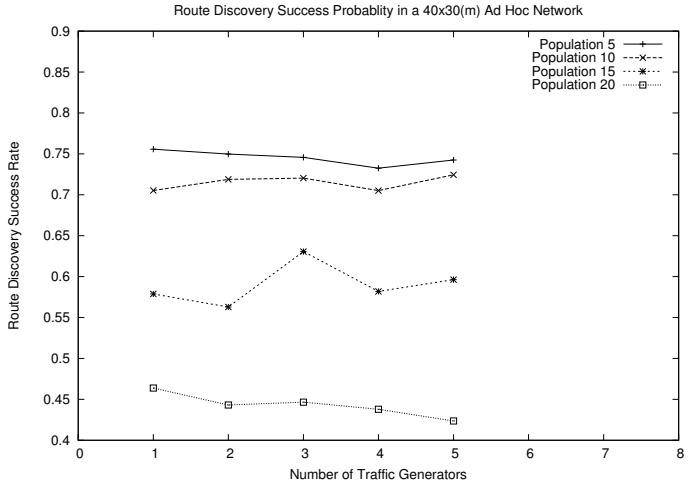


Figure A.1: Route Discovery Success Probability under AVSR at 95% confidence level and 5% precision error.

Results collected form the same simulated network at 99% confidence level and 5% precision level are shown in Figure A.2. The line for population 15 becomes less fluctuated, and a minor trend can be identified. However, we were not stratified with this result and decided to reduce the statistical error of our simulation again by decreasing the precision error.

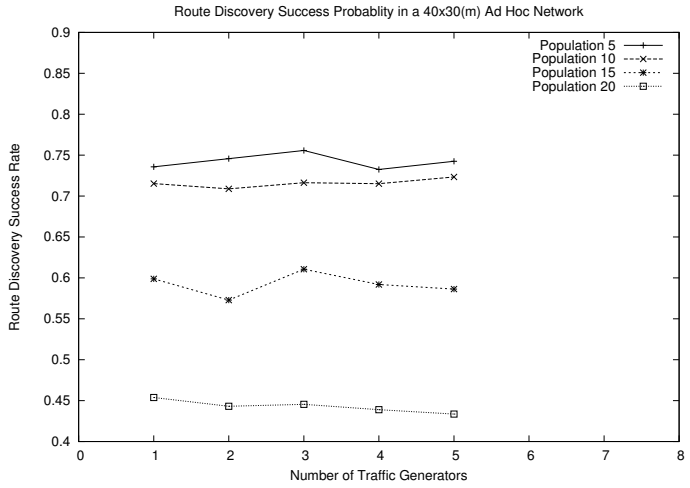


Figure A.2: Route Discovery Success Probability under AVSR at 99% confidence level and 5% precision error.

Figures A.3 presents our final results of this simulation. Results in the simulation were collected with 99%confidence level and 1% precision error. In this figure, all lines are practically flat. Thus we are very confident to claim that RDSP is not dependent on the number of traffic generators in the network.

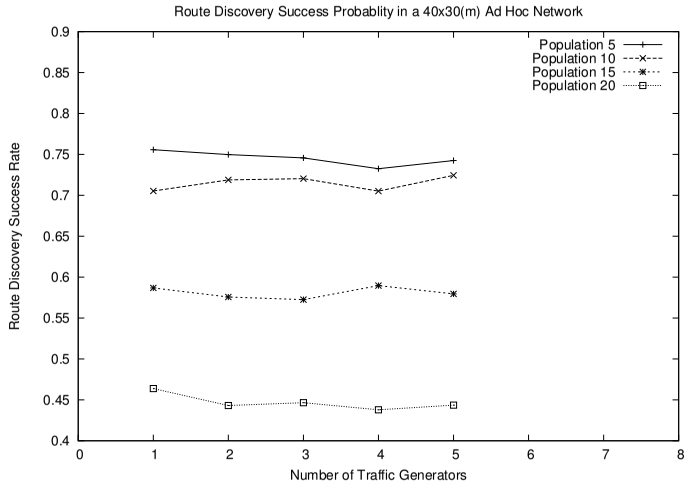


Figure A.3: Route Discovery Success Probability under AVSR at 99% confidence level and 1% precision error.

Bibliography

[1] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A performance comparison of multi-hop ad hoc network routing protocols. In *Proceedings of the Fourth Annual ACM/IEEE Internal Conference on Mobile Computing and Networking (MobiCom'98),* October 25–30, 1998, Dallas, Texas, USA, 1998.

[2] C. Chiang, H. Wu, W. Liu, and M. Gerla. Routing in clustered multihop, mobile wireless networks with fading channel. In *The IEEE Singapore International Conference on Networks*, pages 197–211, 1997.

[3] Thomas Clausen and Philippe Jacquet. *Optimized Link State Routing Protocol.* IETF MANET Working Group, internet draft edition, May 2003.

[4] P. L Ecuyer. Good parameters and implementations for combined multiple recursive random number generator. *Operation Research*, 47(1):1159–164, 1999.

[5] P. L Ecuyer and R. Simard. On the performance of birthday spacing tests with certain families of random number generators. *Mathematics and Computers in Simulation*, 5(1–3):131–137, 2001.

[6] G. Ewing, K. Palikowski, and D. McNickle. *Akaroa 2 User's Manual.* University of Canterbury, Christchurch New Zealand, August 2001.

[7] T. Eyers and H. Schulzrinne. Predicting internet telephony call setup delay. In *Proceeding of 1st IP-Telephony Workshop, Berlin, Germany*, Apri 2000.

[8] Zygmunt J. Haas and Marc R. Pearlman. The performance of query control schemes for the zone routing protocol. In *Proceeding of ACM SIGCOMM98*, pages 167–177, 1998.

[9] Xiaoyan Hong, Kaixin Xu, and Mario Gerla. Scalable routing protocols for mobile ad hoc networks. *IEEE Networks*, pages 11–21, July/August 2002.

[10] David B Johnson and David A Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[11] David B. Johnson, David A. Maltz, Yih-Chun Hu, and Jorjeta G. Jetcheva. *The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR).* IETF MANET Working group, internet–draft edition, Fabuary 2002.

[12] Young-Bae Ko and Nitin H. Vaidya. Location-aided routing (lar) in mobile ad hoc networks. *ACM Mobile Computing and Networking*, pages 66–75, 1998.

[13] Sung-Ju Lee, Julian Hsu, Russel HAyashida, Mario Gerla, and Rajive Bagrodia. Selecting a routing strategy for your ad hoc network. *Computer Communications*, pages 723–733, 2003.

[14] B.M. Leiner, R.J. Ruther, and A.R. Sastry. Goals and challenges of the darpa glomo program [global mobile information systems]. *IEEE Personal Communication*, 3:34–43, December 1996.

[15] Matthias Lott and Bernhard Walke. Performance of a wireless ad hoc network w-chamb. In *Procedings of European Wireless, October, 1999, Munich, Germany*, pages 415–420, 1999.

[16] A. Nasipuri. On demand routing using directional antennas in mobile ad hoc networks. In *Proceeding of the IEEE WCNC 2000.*, 2000.

[17] Julio C. Navas and Tomasz Imielinski. Geocast - geographic addressing and routing. *ACM Mobile Computing and Networking*, pages 66–76, 1997.

[18] Yoram Ofek. Generating a fault–tolerant global clock using high–speed control signals for metanet architecture. *IEEE Transaction on Communication*, 42(5):2179–2188, May 1994.

[19] Vincent D. Park and M. Scott Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proceedings of the Sixteenth Annual*

Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution, pages 1405–1413, 1997.

[20] K. Pawlikowski, H.-D. Joshua Jeong, and J.-S. Ruth Lee. On credibility of simulation studies of telecommunication networks. *IEEE Communications Magazine*, 40(1):132–139, January 2002.

[21] Jörg Peetz. Hiperlan/2 multihop ad hoc communication by multiple-frequency forwarding. In *Proceedings of Vehicular Technology Conference, 2001. VTC 2001 Fall. IEEE VTS 54th, 7–11 October, 2001*, volume 2, pages 2118–2122, 2001.

[22] Guangyu Pei, Mario Gerla, and Tsu-Wei Chen. Fisheye state routing: A routing scheme for ad hoc wireless networks. In *Proceedings of the IEEE International Conference on Communications*, pages 70–74, 2000.

[23] Guangyu Pei, Mario Gerla, Xiaoyan Hong, and Ching-Chuan Chiang. Wireless hierarchical routing protocol with group mobility. In *WNCN99*, September 1999.

[24] Charles Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *Proceedings of the SIGCOMM '94 Conference on Communication Architecture, Protocols and Applications*, pages 234–244, 1994.

[25] Charles E. Perkins, Elizabeth M. Belding-Royer, and Samir R. Das. *Ad hoc On-demand Distance Vector (AODV) Routing.* Mobile Ad Hoc Networking Working Group, internet draft edition, January 1999.

[26] Charles E. Perkins and Elizabeth M. Royer. Ad-hoc on-demand distance vector routing. In *Procedings of the Second IEEE Workahop on Mobile Computing Systems and Applications, 1999*, pages 90–100, 1999.

[27] Amir Qayyum, Laurent Viennot, and Anis Laouiti. Multipoint relaying: An efficient technique for flooding in mobile wireless networks. Technical Report Research Report RR-3898, INRIA, February 2000.

[28] William Stallings. *Data & Computer Communication*, chapter 10. Prentice–Hall Inc., 2000.

[29] C-K Toh. *Ad hoc mobile wireless networks: protocols and systems*, chapter 4. Prentice–Hall Inc., 2002.

[30] Bernhard Walke and Bangnan Xu. On the mac performance of self-organizing broadband multihop multimedaia wireless networks. In *Proceedings of Vehicular Technology Conference, 2001. VTC 2001 Fall. IEEE VTS 54th, 7–11 October, 2001*, volume 2, pages 982–986, 2001.

[31] Bangnan Xu and Bernhard Walke. Design issues of self–organizing broadband wireless networks. *Computer Networks*, 37:73–81, 2001.